


J. E. Gentle
W. Härdle
Y. Mori
(Editors)

Handbook of Computational Statistics

Concepts and Methods

 Springer

Editors

James E. Gentle

George Mason University
Center for Computational Statistics
Fairfax, VA 22030-4444
USA
e-mail: jgentle@gmu.edu

Yuichi Mori

Okayama University of Science
Department of Socio-Information
1-1 Ridai-cho
Okayama 700-0005
Japan
e-mail: mori@soci.ous.ac.jp

Wolfgang Härdle

Humboldt-Universität zu Berlin
Wirtschaftswissenschaftliche Fakultät
Institut für Statistik und Ökonometrie
Spandauer Str. 1
10178 Berlin
Germany
e-mail: haerdle@wiwi.hu-berlin.de

Library of Congress Control Number: 2004106523

ISBN 3-540-40464-3 Springer Berlin Heidelberg New York

This work is subject to copyright. All rights are reserved, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilm or in any other way, and storage in data banks. Duplication of this publication or parts thereof is permitted only under the provisions of the German Copyright Law of September 9, 1965, in its current version, and permission for use must always be obtained from Springer-Verlag. Violations are liable for prosecution under the German Copyright Law.

Springer is a part of Springer Science+Business Media

springeronline.com

© Springer-Verlag Berlin Heidelberg 2004
Printed in Germany

The use of general descriptive names, registered names, trademarks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

Typesetting and production: LE-TeX Jelonek, Schmidt & Vöckler GbR, Leipzig
Cover design and production: deblik, Berlin
Printed on acid-free paper 40/3142/YL 5 4 3 2 1 0

Table of Contents

I. Computational Statistics

I.1 Computational Statistics: An Introduction <i>James E. Gentle, Wolfgang Härdle, Yuichi Mori</i>	3
--	---

II. Statistical Computing

II.1 Basic Computational Algorithms <i>John Monahan</i>	19
II.2 Random Number Generation <i>Pierre L'Ecuyer</i>	35
II.3 Markov Chain Monte Carlo Technology <i>Siddhartha Chib</i>	71
II.4 Numerical Linear Algebra <i>Lenka Čížková, Pavel Čížek</i>	103
II.5 The EM Algorithm <i>Shu Kay Ng, Thriyambakam Krishnan, Geoffrey J. McLachlan</i>	137
II.6 Stochastic Optimization <i>James C. Spall</i>	169
II.7 Transforms in Statistics <i>Branislav Vidakovic</i>	199
II.8 Parallel Computing Techniques <i>Junji Nakano</i>	237
II.9 Statistical Databases <i>Claus Boyens, Oliver Günther, Hans-J. Lenz</i>	267
II.10 Interactive and Dynamic Graphics <i>Jürgen Symanzik</i>	293

II.11 The Grammar of Graphics	
<i>Leland Wilkinson</i>	337
II.12 Statistical User Interfaces	
<i>Sigbert Klinke</i>	379
II.13 Object Oriented Computing	
<i>Miroslav Virius</i>	403

III. Statistical Methodology

III.1 Model Selection	
<i>Yuedong Wang</i>	437
III.2 Bootstrap and Resampling	
<i>Enno Mammen, Swagata Nandi</i>	467
III.3 Design and Analysis of Monte Carlo Experiments	
<i>Jack P.C. Kleijnen</i>	497
III.4 Multivariate Density Estimation and Visualization	
<i>David W. Scott</i>	517
III.5 Smoothing: Local Regression Techniques	
<i>Catherine Loader</i>	539
III.6 Dimension Reduction Methods	
<i>Masahiro Mizuta</i>	565
III.7 Generalized Linear Models	
<i>Marlene Müller</i>	591
III.8 (Non) Linear Regression Modeling	
<i>Pavel Čížek</i>	621
III.9 Robust Statistics	
<i>Laurie Davies, Ursula Gather</i>	655
III.10 Semiparametric Models	
<i>Joel L. Horowitz</i>	697
III.11 Bayesian Computational Methods	
<i>Christian P. Robert</i>	719
III.12 Computational Methods in Survival Analysis	
<i>Toshinari Kamakura</i>	767
III.13 Data and Knowledge Mining	
<i>Adalbert Wilhelm</i>	787
III.14 Recursive Partitioning and Tree-based Methods	
<i>Heping Zhang</i>	813
III.15 Support Vector Machines	
<i>Sebastian Mika, Christin Schäfer, Pavel Laskov, David Tax,</i> <i>Klaus-Robert Müller</i>	841
III.16 Bagging, Boosting and Ensemble Methods	
<i>Peter Bühlmann</i>	877

IV. Selected Applications

IV.1 Computationally Intensive Value at Risk Calculations	
<i>Rafat Weron</i>	911
IV.2 Econometrics	
<i>Luc Bauwens, Jeroen V.K. Rombouts</i>	951
IV.3 Statistical and Computational Geometry of Biomolecular Structure	
<i>Iosif I. Vaisman</i>	981
IV.4 Functional Magnetic Resonance Imaging	
<i>William F. Eddy, Rebecca L. McNamee</i>	1001
IV.5 Network Intrusion Detection	
<i>David J. Marchette</i>	1029
Subject Index	1053

List of Contributors

Luc Bauwens

Université catholique de Louvain
CORE and Department of Economics
Belgium
bauwens@core.ucl.ac.be

Lenka Čížková

Czech Technical University in Prague
Faculty of Nuclear Sciences
and Physical Engineering
The Czech Republic
lenka_cizkova@web.de

Claus Boyens

Humboldt-Universität zu Berlin
Institut für Wirtschaftsinformatik
Wirtschaftswissenschaftliche Fakultät
Germany

Laurie Davies

University of Essen
Department of Mathematics
Germany
laurie.davies@uni-essen.de

Peter Bühlmann

ETH Zürich
Seminar für Statistik
Switzerland
buhlmann@stat.math.ethz.ch

William F. Eddy

Carnegie Mellon University
Department of Statistics
USA
bill@stat.cmu.edu

Siddhartha Chib

Washington University in Saint Louis
John M. Olin School of Business
chib@wustl.edu

Ursula Gather

University of Dortmund
Department of Statistics
Germany
gather@statistik.uni-dortmund.de

Pavel Čížek

Tilburg University
Department of Econometrics &
Operations Research
The Netherlands
P.Cizek@uvt.nl

James E. Gentle

George Mason University
USA
jgentle@gmu.edu

Oliver Günther

Humboldt-Universität zu Berlin
Institut für Wirtschaftsinformatik
Wirtschaftswissenschaftliche Fakultät
Germany

Wolfgang Härdle

Humboldt-Universität zu Berlin
Wirtschaftswissenschaftliche Fakultät
Institut für Statistik und Ökonometrie
Germany
haerdle@wiwi.hu-berlin.de

Joel L. Horowitz

Northwestern University
Department of Economics
USA

Toshinari Kamakura

Chuo University
Japan
kamakura@indsys.chuo-u.ac.jp

Jack P.C. Kleijnen

Tilburg University
Department of Information Systems
and Management
Center for Economic Research (CentER)
The Netherlands
Kleijnen@uvt.nl

Sigbert Klinke

Humboldt-Universität zu Berlin
Wirtschaftswissenschaftliche Fakultät
Institut für Statistik und Ökonometrie
Germany
sigbert@wiwi.hu-berlin.de

Thriyambakam Krishnan

Systat Software Asia-Pacific Ltd.
Bangalore
India
krishnan@systat.com

Pavel Laskov

Fraunhofer FIRST
Department IDA
Germany
laskov@first.fhg.de

Pierre L'Ecuyer

Université de Montréal
GERAD and
Département d'informatique
et de recherche opérationnelle
Canada

Hans-J. Lenz

Freie Universität Berlin
Fachbereich Wirtschaftswissenschaft
Institut für Produktion,
Wirtschaftsinformatik
und Operations Research und
Institut für Statistik und Ökonometrie
Germany

Catherine Loader

Case Western Reserve University
Department of Statistics
USA
catherine@case.edu

Enno Mammen

University of Mannheim
Department of Economics
Germany
emammen@rumms.uni-mannheim.de

David J. Marchette

John Hopkins University
Whiting School of Engineering
USA
dmarche@nswc.navy.mil

Geoffrey J. McLachlan

University of Queensland

Department of Mathematics
Australia
gjm@maths.uq.edu.au

Rebecca L. McNamee
University of Pittsburgh
USA
rlandes@stat.cmu.edu

Sebastian Mika
idalab GmbH
Germany
mika@idalab.de
and
Fraunhofer FIRST
Department IDA
Germany
mika@first.fhg.de

Masahiro Mizuta
Hokkaido University
Information Initiative Center
Japan
mizuta@cims.hokudai.ac.jp

John Monahan
North Carolina State University
Department of Statistics
USA
monahan@stat.ncsu.edu

Yuichi Mori
Okayama University of Science
Department of Socioinformation
Japan
mori@soci.ous.ac.jp

Klaus-Robert Müller
Fraunhofer FIRST
Department IDA
Germany
klaus@first.fhg.de

and
University Potsdam
Department of Computer Science
Germany

Marlene Müller
Fraunhofer ITWM
Germany
marlene.mueller@gmx.de

Junji Nakano
The Institute
of Statistical Mathematics
Japan
nakanoj@ism.ac.jp

Swagata Nandi
University Heidelberg
Institute of Applied Mathematics
Germany
nandi@statlab.uni-heidelberg.de

Shu Kay Ng
University of Queensland
Department of Mathematics
Australia
skn@maths.uq.edu.au

Christian P. Robert
Université Paris Dauphine
CERMADE
France
christian.robert
@ceremade.dauphine.fr

Jeroen V.K. Rombouts
Université catholique de Louvain
CORE and Department of Economics
Belgium
rombouts@core.ucl.ac.be

Christin Schäfer
Fraunhofer FIRST
Department IDA
Germany
christin@first.fhg.de

David W. Scott
Rice University
Department of Statistics
USA
scottdw@rice.edu

James C. Spall
The Johns Hopkins University
Applied Physics Laboratory
USA
james.spall@jhuapl.edu

Jürgen Symanzik
Utah State University
Department of Mathematics
and Statistics
USA
symanzik@math.usu.edu

David Tax
Delft University of Technology
The Netherlands

Iosif I. Vaisman
George Mason University
School of Computational Sciences
USA
ivaisman@gmu.edu

Brani Vidakovic
School of Industrial
and Systems Engineering
Georgia Institute of Technology

USA
brani@isye.gatech.edu

Miroslav Virius
Czech Technical University in Prague
Faculty of Nuclear Sciences
and Physical Engineering
Czech Republic
virius@kml.fjfi.cvut.cz

Yuedong Wang
University of California
Department of Statistics
and Applied Probability
USA
yuedong@pstat.ucsb.edu

Rafał Weron
Hugo Steinhaus Center
for Stochastic Methods
Wrocław University of Technology
Poland
rweron@im.pwr.wroc.pl

Adalbert Wilhelm
International University Bremen
School of Humanities
and Social Sciences
Germany
a.wilhelm@iu-bremen.de

Leland Wilkinson
SPSS Inc. and Northwestern University
USA
leland@spss.com

Heping Zhang
Yale University School of Medicine
Department of Epidemiology
and Public Health
USA
heping.zhang@yale.edu

Part I
Computational Statistics



Computational Statistics: An Introduction

I.1

James E. Gentle, Wolfgang Härdle, Yuichi Mori

1.1	<i>Computational Statistics and Data Analysis</i>	4
1.2	<i>The Emergence of a Field of Computational Statistics</i>	6
	Early Developments in Statistical Computing.....	7
	Early Conferences and Formation of Learned Societies	7
	The PC.....	8
	The Cross Currents of Computational Statistics	9
	Literature	9
1.3	<i>Why This Handbook</i>	11
	Summary and Overview; Part II: Statistical Computing.....	11
	Summary and Overview; Part III: Statistical Methodology	13
	Summary and Overview; Part IV: Selected Applications	14
	The Ehandbook.....	15
	Future Handbooks in Computational Statistics.....	15

Computational Statistics and Data Analysis

1.1

To do data analysis is to do computing. Statisticians have always been heavy users of whatever computing facilities are available to them. As the computing facilities have become more powerful over the years, those facilities have obviously decreased the amount of effort the statistician must expend to do routine analyses. As the computing facilities have become more powerful, an opposite result has occurred, however; the computational aspect of the statistician's work has increased. This is because of paradigm shifts in statistical analysis that are enabled by the computer.

Statistical analysis involves use of observational data together with domain knowledge to develop a model to study and understand a data-generating process. The data analysis is used to refine the model or possibly to select a different model, to determine appropriate values for terms in the model, and to use the model to make inferences concerning the process. This has been the paradigm followed by statisticians for centuries. The advances in statistical theory over the past two centuries have not changed the paradigm, but they have improved the specific methods. The advances in computational power have enabled newer and more complicated statistical methods. Not only has the exponentially-increasing computational power allowed use of more detailed and better models, however, it has shifted the paradigm slightly. Many alternative views of the data can be examined. Many different models can be explored. Massive amounts of simulated data can be used to study the model/data possibilities.

When exact models are mathematically intractable, approximate methods, which are often based on asymptotics, or methods based on estimated quantities must be employed. Advances in computational power and developments in theory have made *computational inference* a viable and useful alternative to the standard methods of asymptotic inference in traditional statistics. Computational inference is based on simulation of statistical models.

The ability to perform large numbers of computations almost instantaneously and to display graphical representations of results immediately has opened many new possibilities for statistical analysis. The hardware and software to perform these operations are readily available and are accessible to statisticians with no special expertise in computer science. This has resulted in a two-way feedback between statistical theory and statistical computing. The advances in statistical computing suggest new methods and development of supporting theory; conversely, the advances in theory and methods necessitate new computational methods.

Computing facilitates the development of statistical theory in two ways. One way is the use of symbolic computational packages to help in mathematical derivations (particularly in reducing the occurrences of errors in going from one line to the next!). The other way is in the quick exploration of promising (or unpromising!) methods by simulations. In a more formal sense also, simulations allow evaluation and comparison of statistical methods under various alternatives. This is a widely-used research method. For example, out of 61 articles published in the *Theory and*

Methods section of the *Journal of the American Statistical Association* in 2002, 50 reported on Monte Carlo studies of the performance of statistical methods. A general outline of many research articles in statistics is

1. state the problem and summarize previous work on it,
2. describe a new approach,
3. work out some asymptotic properties of the new approach,
4. conduct a Monte Carlo study showing the new approach in a favorable light.

Much of the effort in mathematical statistics has been directed toward the easy problems of exploration of asymptotic properties. The harder problems for finite samples require different methods. Carefully conducted and reported Monte Carlo studies often provide more useful information on the relative merits of statistical methods in finite samples from a range of model scenarios.

While to do data analysis is to compute, we do not identify all data analysis, which necessarily uses the computer, as “statistical computing” or as “computational statistics”. By these phrases we mean something more than just using a statistical software package to do a standard analysis. We use the term “statistical computing” to refer to the computational methods that enable statistical methods. Statistical computing includes numerical analysis, database methodology, computer graphics, software engineering, and the computer/human interface. We use the term “computational statistics” somewhat more broadly to include not only the methods of statistical computing, but also statistical methods that are computationally intensive. Thus, to some extent, “computational statistics” refers to a large class of modern statistical methods. Computational statistics is grounded in mathematical statistics, statistical computing, and applied statistics. While we distinguish “computational statistics” from “statistical computing”, the emergence of the field of computational statistics was coincidental with that of statistical computing, and would not have been possible without the developments in statistical computing.

One of the most significant results of the developments in statistical computing during the past few decades has been the statistical software package. There are several of these, but a relatively small number that are in widespread use. While referees and editors of scholarly journals determine what statistical theory and methods are *published*, the developers of the major statistical software packages determine what statistical methods are *used*. Computer programs have become necessary for statistical analysis. The specific methods of a statistical analysis are often determined by the available software. This, of course, is not a desirable situation, but, ideally, the two-way feedback between statistical theory and statistical computing diminishes the effect over time.

The importance of computing in statistics is also indicated by the fact that there are at least ten major journals with titles that contain some variants of both “computing” and “statistics”. The journals in the mainstream of statistics without “computing” in their titles also have a large proportion of articles in the fields of statistical computing and computational statistics. This is because, to a large extent, recent developments in statistics and in the computational sciences have

gone hand in hand. There are also two well-known learned societies with a primary focus in statistical computing: the International Association for Statistical Computing (IASC), which is an affiliated society of the International Statistical Institute (ISI), and the Statistical Computing Section of the American Statistical Association (ASA). There are also a number of other associations focused on statistical computing and computational statistics, such as the Statistical Computing Section of the Royal Statistical Society (RSS), and the Japanese Society of Computational Statistics (JSCS).

Developments in computing and the changing role of computations in statistical work have had significant effects on the curricula of statistical education programs both at the graduate and undergraduate levels. Training in statistical computing is a major component in some academic programs in statistics (see Gentle, 2004, Lange, 2004, and Monahan, 2004). In all academic programs, some amount of computing instruction is necessary if the student is expected to work as a statistician. The extent and the manner of integration of computing into an academic statistics program, of course, change with the developments in computing hardware and software and advances in computational statistics.

We mentioned above the two-way feedback between statistical theory and statistical computing. There is also an important two-way feedback between applications and statistical computing, just as there has always been between applications and any aspect of statistics. Although data scientists seek commonalities among methods of data analysis, different areas of application often bring slightly different problems for the data analyst to address. In recent years, an area called “data mining” or “knowledge mining” has received much attention. The techniques used in data mining are generally the methods of exploratory data analysis, of clustering, and of statistical learning, applied to very large and, perhaps, diverse datasets. Scientists and corporate managers alike have adopted data mining as a central aspect of their work. Specific areas of application also present interesting problems to the computational statistician. Financial applications, particularly risk management and derivative pricing, have fostered advances in computational statistics. Biological applications, such as bioinformatics, microarray analysis, and computational biology, are fostering increasing levels of interaction with computational statistics.

The hallmarks of computational statistics are the use of more complicated models, larger datasets with both more observations and more variables, unstructured and heterogeneous datasets, heavy use of visualization, and often extensive simulations.

The Emergence of a Field of Computational Statistics

Statistical computing is truly a multidisciplinary field and the diverse problems have created a yeasty atmosphere for research and development. This has been the

case from the beginning. The roles of statistical laboratories and the applications that drove early developments in statistical computing are surveyed by Grier (1999). As digital computers began to be used, the field of statistical computing came to embrace not only numerical methods but also a variety of topics from computer science.

The development of the field of statistical computing was quite fragmented, with advances coming from many directions – some by persons with direct interest and expertise in computations, and others by persons whose research interests were in the applications, but who needed to solve a computational problem. Through the 1950s the major facts relevant to statistical computing were scattered through a variety of journal articles and technical reports. Many results were incorporated into computer programs by their authors and never appeared in the open literature. Some persons who contributed to the development of the field of statistical computing were not aware of the work that was beginning to put numerical analysis on a sound footing. This hampered advances in the field.

Early Developments in Statistical Computing

1.2.1

An early book that assembled much of the extant information on digital computations in the important area of linear computations was by Dwyer (1951). In the same year, Von Neumann's (1951) NBS publication described techniques of random number generation and applications in Monte Carlo. At the time of these publications, however, access to digital computers was not widespread. Dwyer (1951) was also influential in regression computations performed on calculators. Some techniques, such as use of "machine formulas", persisted into the age of digital computers.

Developments in statistical computing intensified in the 1960s, as access to digital computers became more widespread. Grier (1991) describes some of the effects on statistical practice by the introduction of digital computers, and how statistical applications motivated software developments. The problems of rounding errors in digital computations were discussed very carefully in a pioneering book by Wilkinson (1963). A number of books on numerical analysis using digital computers were beginning to appear. The techniques of random number generation and Monte Carlo were described by Hammersley and Handscomb (1964). In 1967 the first book specifically on statistical computing appeared, Hemmerle (1967).

Early Conferences and Formation of Learned Societies

1.2.2

The 1960s also saw the beginnings of conferences on statistical computing and sections on statistical computing within the major statistical societies. The Royal Statistical Society sponsored a conference on statistical computing in December 1966. The papers from this conference were later published in the RSS's *Applied Statistics* journal. The conference led directly to the formation of a Working Party on Statistical Computing within the Royal Statistical Society. The first Symposium on the Interface of Computer Science and Statistics was held February 1,

1967. This conference has continued as an annual event with only a few exceptions since that time (see Goodman, 1993, Billard and Gentle, 1993, and Wegman, 1993). The attendance at the Interface Symposia initially grew rapidly year by year and peaked at over 600 in 1979. In recent years the attendance has been slightly under 300. The proceedings of the Symposium on the Interface have been an important repository of developments in statistical computing. In April, 1969, an important conference on statistical computing was held at the University of Wisconsin. The papers presented at that conference were published in a book edited by Milton and Nelder (1969), which helped to make statisticians aware of the useful developments in computing and of their relevance to the work of applied statisticians.

In the 1970s two more important societies devoted to statistical computing were formed. The Statistical Computing Section of the ASA was formed in 1971 (see Chambers and Ryan, 1990). The Statistical Computing Section organizes sessions at the annual meetings of the ASA, and publishes proceedings of those sessions. The International Association for Statistical Computing (IASC) was founded in 1977 as a Section of ISI. In the meantime, the first of the biennial COMPSTAT Conferences on computational statistics was held in Vienna in 1974. Much later, regional sections of the IASC were formed, one in Europe and one in Asia. The European Regional Section of the IASC is now responsible for the organization of the COMPSTAT conferences.

Also, beginning in the late 1960s and early 1970s, most major academic programs in statistics offered one or more courses in statistical computing. More importantly, perhaps, instruction in computational techniques has permeated many of the standard courses in applied statistics.

As mentioned above, there are several journals whose titles include some variants of both “computing” and “statistics”. The first of these, the *Journal of Statistical Computation and Simulation*, was begun in 1972. There are dozens of journals in numerical analysis and in areas such as “computational physics”, “computational biology”, and so on, that publish articles relevant to the fields of statistical computing and computational statistics.

By 1980 the field of statistical computing, or computational statistics, was well-established as a distinct scientific subdiscipline. Since then, there have been regular conferences in the field, there are scholarly societies devoted to the area, there are several technical journals in the field, and courses in the field are regularly offered in universities.

1.2.3 The PC

The 1980s was a period of great change in statistical computing. The personal computer brought computing capabilities to almost everyone. With the PC came a change not only in the number of participants in statistical computing, but, equally important, completely different attitudes toward computing emerged. Formerly, to do computing required an account on a mainframe computer. It required laboriously entering arcane computer commands onto punched cards, taking these

cards to a card reader, and waiting several minutes or perhaps a few hours for some output – which, quite often, was only a page stating that there was an error somewhere in the program. With a personal computer for the exclusive use of the statistician, there was no incremental costs for running programs. The interaction was personal, and generally much faster than with a mainframe. The software for PCs was friendlier and easier to use. As might be expected with many non-experts writing software, however, the general quality of software probably went down.

The democratization of computing resulted in rapid growth in the field, and rapid growth in software for statistical computing. It also contributed to the changing paradigm of the data sciences.

The Cross Currents of Computational Statistics

1.2.4

Computational statistics of course is more closely related to statistics than to any other discipline, and computationally-intensive methods are becoming more commonly used in various areas of application of statistics. Developments in other areas, such as computer science and numerical analysis, are also often directly relevant to computational statistics, and the research worker in this field must scan a wide range of literature.

Numerical methods are often developed in an ad hoc way, and may be reported in the literature of any of a variety of disciplines. Other developments important for statistical computing may also be reported in a wide range of journals that statisticians are unlikely to read. Keeping abreast of relevant developments in statistical computing is difficult not only because of the diversity of the literature, but also because of the interrelationships between statistical computing and computer hardware and software.

An example of an area in computational statistics in which significant developments are often made by researchers in other fields is Monte Carlo simulation. This technique is widely used in all areas of science, and researchers in various areas often contribute to the development of the science and art of Monte Carlo simulation. Almost any of the methods of Monte Carlo, including random number generation, are important in computational statistics.

Literature

1.2.5

Some of the major periodicals in statistical computing and computational statistics are the following. Some of these journals and proceedings are refereed rather rigorously, some refereed less so, and some are not refereed.

- *ACM Transactions on Mathematical Software*, published quarterly by the ACM (Association for Computing Machinery), includes algorithms in Fortran and C. Most of the algorithms are available through `netlib`. The ACM collection of algorithms is sometimes called *CALGO*.
www.acm.org/toms/

- *ACM Transactions on Modeling and Computer Simulation*, published quarterly by the ACM.
www.acm.org/tomacs/
- *Applied Statistics*, published quarterly by the Royal Statistical Society. (Until 1998, it included algorithms in Fortran. Some of these algorithms, with corrections, were collected by Griffiths and Hill, 1985. Most of the algorithms are available through `statlib` at Carnegie Mellon University.)
www.rss.org.uk/publications/
- *Communications in Statistics – Simulation and Computation*, published quarterly by Marcel Dekker. (Until 1996, it included algorithms in Fortran. Until 1982, this journal was designated as *Series B*.)
www.dekker.com/servlet/product/productid/SAC/
- *Computational Statistics* published quarterly by Physica-Verlag (formerly called *Computational Statistics Quarterly*).
comst.wiwi.hu-berlin.de/
- *Computational Statistics. Proceedings of the xx-th Symposium on Computational Statistics (COMPSTAT)*, published biennially by Physica-Verlag/Springer.
- *Computational Statistics & Data Analysis*, published by Elsevier Science. There are twelve issues per year. (This is also the official journal of the International Association for Statistical Computing and as such incorporates the *Statistical Software Newsletter*.)
www.cbs.nl/isi/csda.htm
- *Computing Science and Statistics*. This is an annual publication containing papers presented at the Interface Symposium. Until 1992, these proceedings were named *Computer Science and Statistics: Proceedings of the xx-th Symposium on the Interface*. (The 24th symposium was held in 1992.) In 1997, Volume 29 was published in two issues: Number 1, which contains the papers of the regular Interface Symposium; and Number 2, which contains papers from another conference. The two numbers are not sequentially paginated. Since 1999, the proceedings have been published only in CD-ROM form, by the Interface Foundation of North America.
www.galaxy.gmu.edu/stats/IFNA.html
- *Journal of Computational and Graphical Statistics*, published quarterly as a joint publication of ASA, the Institute of Mathematical Statistics, and the Interface Foundation of North America.
www.amstat.org/publications/jcgs/
- *Journal of the Japanese Society of Computational Statistics*, published once a year by JSCS.
www.jscs.or.jp/oubun/indexE.html
- *Journal of Statistical Computation and Simulation*, published in twelve issues per year by Taylor & Francis.
www.tandf.co.uk/journals/titles/00949655.asp
- *Proceedings of the Statistical Computing Section*, published annually by ASA.
www.amstat.org/publications/

- *SIAM Journal on Scientific Computing*, published bimonthly by SIAM. This journal was formerly *SIAM Journal on Scientific and Statistical Computing*.
www.siam.org/journals/sisc/sisc.htm
- *Statistical Computing & Graphics Newsletter*, published quarterly by the Statistical Computing and the Statistical Graphics Sections of ASA.
www.statcomputing.org/
- *Statistics and Computing*, published quarterly by Chapman & Hall.

In addition to literature and learned societies in the traditional forms, an important source of communication and a repository of information are computer databases and forums. In some cases, the databases duplicate what is available in some other form, but often the material and the communications facilities provided by the computer are not available elsewhere.

Why This Handbook

1.3

The purpose of this handbook is to provide a survey of the basic concepts of computational statistics; that is, *Concepts and Fundamentals*. A glance at the table of contents reveals a wide range of articles written by experts in various subfields of computational statistics. The articles are generally expository, taking the reader from the basic concepts to the current research trends. The emphasis throughout, however, is on the concepts and fundamentals. Most chapters have extensive and up-to-date references to the relevant literature (with, in many cases, perhaps a perponderance of self-references!)

We have organized the topics into Part II on “statistical computing”, that is, the computational methodology, and Part III “statistical methodology”, that is, the techniques of applied statistics that are computer-intensive, or otherwise make use of the computer as a tool of discovery, rather than as just a large and fast calculator. The final part of the handbook covers a number of application areas in which computational statistics plays a major role are surveyed.

Summary and Overview; Part II: Statistical Computing

1.3.1

The thirteen chapters of Part II, Statistical Computing, cover areas of numerical analysis and computer science or informatics that are relevant for statistics. These areas include computer arithmetic, algorithms, database methodology, languages and other aspects of the user interface, and computer graphics.

In the first chapter of this part, Monahan describes how numbers are stored on the computer, how the computer does arithmetic, and more importantly what the implications are for statistical (or other) computations. In this relatively short chapter, he then discusses some of the basic principles of numerical algorithms, such as divide and conquer. Although many statisticians do not need to know the details, it is important that all statisticians understand the implications of

computations within a system of numbers and operators that is not the same system that we are accustomed to in mathematics. Anyone developing computer algorithms, no matter how trivial the algorithm may appear, must understand the details of the computer system of numbers and operators.

One of the important uses of computers in statistics, and one that is central to computational statistics, is the simulation of random processes. This is a theme we will see in several chapters of this handbook. In Part II, the basic numerical methods relevant to simulation are discussed. First, L'Ecuyer describes the basics of random number generation, including assessing the quality of random number generators, and simulation of random samples from various distributions. Next Chib describes one special use of computer-generated random numbers in a class of methods called Markov chain Monte Carlo. These two chapters describe the basic numerical methods used in computational inference. Statistical methods using simulated samples are discussed further in Part III.

The next four chapters of Part II address specific numerical methods. The first of these, methods for linear algebraic computations, are discussed by Čížková and Čížek. These basic methods are used in almost all statistical computations. Optimization is another basic method used in many statistical applications. Chapter II.5 on the EM algorithm and its variations by Ng, Krishnan, and McLachlan, and Chap. II.6 on stochastic optimization by Spall address two specific areas of optimization. Finally, in Chap. II.7, Vidakovic discusses transforms that effectively restructure a problem by changing the domain. These transforms are statistical functionals, the most well-known of which are Fourier transforms and wavelet transforms.

The next two chapters focus on efficient usage of computing resources. For numerically-intensive applications, parallel computing is both the most efficient and the most powerful approach. In Chap. II.8 Nakano describes for us the general principles, and then some specific techniques for parallel computing. Understanding statistical databases is important not only because of the enhanced efficiency that appropriate data structures allow in statistical computing, but also because of the various types of databases the statistician may encounter in data analysis. In Chap. II.9 on statistical databases, Boyens, Günther, and Lenz give us an overview of the basic design issues and a description of some specific database management systems.

The next two chapters are on statistical graphics. The first of these chapters, by Symanzik, spans our somewhat artificial boundary of Part II (statistical computing) and Part III (statistical methodology, the real heart and soul of computational statistics). This chapter covers some of the computational details, but also addresses the usage of interactive and dynamic graphics in data analysis. Wilkinson, in Chap. II.11, describes a paradigm, the grammar of graphics, for developing and using systems for statistical graphics.

In order for statistical software to be usable and useful, it must have a good user interface. In Chap. II.12 on statistical user interfaces, Klinke discusses some of the general design principles of a good user interface and describes some interfaces that are implemented in current statistical software packages. In the development and

use of statistical software, an object oriented approach provides a consistency of design and allows for easier software maintenance and the integration of software developed by different people at different times. Virius discusses this approach in the final chapter of Part II, on object oriented computing.

Summary and Overview; Part III: Statistical Methodology

1.3.2

Part III covers several aspects of computational statistics. In this part the emphasis is on the statistical methodology that is enabled by computing. Computers are useful in all aspects of statistical data analysis, of course, but in Part III, and generally in computational statistics, we focus on statistical methods that are computationally intensive. Although a theoretical justification of these methods often depends on asymptotic theory, in particular, on the asymptotics of the empirical cumulative distribution function, asymptotic inference is generally replaced by computational inference.

The first three chapters of this part deal directly with techniques of computational inference; that is, the use of cross validation, resampling, and simulation of data-generating processes to make decisions and to assign a level of confidence to the decisions. Wang opens Part III with a discussion of model choice. Selection of a model implies consideration of more than one model. As we suggested above, this is one of the hallmarks of computational statistics: looking at data through a variety of models. Wang begins with the familiar problem of variable selection in regression models, and then moves to more general problems in model selection. Cross validation and generalizations of that method are important techniques for addressing the problems. Next, in Chap. III.2 Mammen and Nandi discuss a class of resampling techniques that have wide applicability in statistics, from estimating variances and setting confidence regions to larger problems in statistical data analysis. Computational inference depends on simulation of data-generating processes. Any such simulation is an *experiment*. In the third chapter of Part III, Kleijnen discusses principles for design and analysis of experiments using computer models.

In Chap. III.4, Scott considers the general problem of estimation of a multivariate probability density function. This area is fundamental in statistics, and it utilizes several of the standard techniques of computational statistics, such as cross validation and visualization methods.

The next four chapters of Part III address important issues for discovery and analysis of relationships among variables. First, Loader discusses local smoothing using a variety of methods, including kernels, splines, and orthogonal series. Smoothing is fitting of asymmetric models, that is, models for the effects of a given set of variables (“independent variables”) on another variable or set of variables. The methods of Chap. III.5 are generally nonparametric, and will be discussed from a different standpoint in Chap. III.10. Next, in Chap. III.6 Mizuta describes ways of using the relationships among variables to reduce the effective dimensionality

of a problem. The next two chapters return to the use of asymmetric models: Müller discusses generalized linear models, and Čížek describes computational and inferential methods for dealing with nonlinear regression models.

In Chap. III.9, Gather and Davies discuss various issues of robustness in statistics. Robust methods are important in such applications as those in financial modeling, discussed in Chap. IV.2. One approach to robustness is to reduce the dependence on parametric assumptions. Horowitz, in Chap. III.10, describes semi-parametric models that make fewer assumptions about the form.

One area in which computational inference has come to play a major role is in Bayesian analysis. Computational methods have enabled a Bayesian approach in practical applications, because no longer is this approach limited to simple problems or conjugate priors. Robert, in Chap. III.11, describes ways that computational methods are used in Bayesian analyses.

Survival analysis, with applications in both medicine and product reliability, has become more important in recent years. Kamakura, in Chap. III.12, describes various models used in survival analysis and the computational methods for analyzing such models.

The final four chapters of Part III address an exciting area of computational statistics. The general area may be called “data mining”, although this term has a rather anachronistic flavor because of the hype of the mid-1990s. Other terms such as “knowledge mining” or “knowledge discovery in databases” (“KDD”) are also used. To emphasize the roots in artificial intelligence, which is a somewhat discredited area, the term “computational intelligence” is also used. This is an area in which machine learning from computer science and statistical learning have merged. In Chap. III.13 Wilhelm provides an introduction and overview of data and knowledge mining, as well as a discussion of some of the vagaries of the terminology as researchers have attempted to carve out a field and to give it scientific legitimacy. Subsequent chapters describe specific methods for statistical learning: Zhang discusses recursive partitioning and tree based methods; Mika, Schäfer, Laskov, Tax, and Müller discuss support vector machines; and Bühlmann describes various ensemble methods.

Summary and Overview; Part IV: Selected Applications

1.3.3

Finally, in Part IV, there are five chapters on various applications of computational statistics. The first, by Weron, discusses stochastic modeling of financial data using heavy-tailed distributions. Next, in Chap. IV.2 Bauwens and Rombouts describe some problems in economic data analysis and computational statistical methods to address them. Some of the problems, such as nonconstant variance, discussed in this chapter on econometrics are also important in finance.

Human biology has become one of the most important areas of application, and many computationally-intensive statistical methods have been developed, refined, and brought to bear on problems in this area. First, Vaisman describes approaches

to understanding the geometrical structure of protein molecules. While much is known about the order of the components of the molecules, the three-dimensional structure for most important protein molecules is not known, and the tools for discovery of this structure need extensive development. Next, Eddy and McNamee describe some statistical techniques for analysis of MRI data. The important questions involve the functions of the various areas in the brain. Understanding these will allow more effective treatment of diseased or injured areas and the resumption of more normal activities by patients with neurological disorders.

Finally, Marchette discusses statistical methods for computer network intrusion detection. Because of the importance of computer networks around the world, and because of their vulnerability to unauthorized or malicious intrusion, detection has become one of the most important – and interesting – areas for data mining.

The articles in this handbook cover the important subareas of computational statistics and give some flavor of the wide range of applications. While the articles emphasize the basic concepts and fundamentals of computational statistics, they provide the reader with tools and suggestions for current research topics. The reader may turn to a specific chapter for background reading and references on a particular topic of interest, but we also suggest that the reader browse and ultimately peruse articles on unfamiliar topics. Many surprising and interesting tidbits will be discovered!

The Ehandbook

1.3.4

A unique feature of this handbook is the supplemental ebook format. Our ebook design offers a HTML file with links to world wide computing servers. This HTML version can be downloaded onto a local computer via a licence card included in this handbook.

Future Handbooks in Computational Statistics

1.3.5

This handbook on concepts and fundamentals sets the stage for future handbooks that go more deeply into the various subfields of computational statistics. These handbooks will each be organized around either a specific class of theory and methods, or else around a specific area of application.

The development of the field of computational statistics has been rather fragmented. We hope that the articles in this handbook series can provide a more unified framework for the field.

References

Billard, L. and Gentle, J.E. (1993). The middle years of the Interface, *Computing Science and Statistics*, 25:19–26.

- Chambers, J.M. and Ryan, B.F. (1990). The ASA Statistical Computing Section, *The American Statistician*, 44(2):87–89.
- Dwyer, P.S. (1951), *Linear Computations*, John Wiley and Sons, New York.
- Gentle, J.E. (2004). Courses in statistical computing and computational statistics, *The American Statistician*, 58:2–5.
- Goodman, A. (1993). Interface insights: From birth into the next century, *Computing Science and Statistics*, 25:14–18.
- Grier, D.A. (1991). Statistics and the introduction of digital computers, *Chance*, 4(3):30–36.
- Grier, D.A. (1999), Statistical laboratories and the origins of statistical computing, *Chance*, 4(2):14–20.
- Hammersley, J.M. and Handscomb, D.C. (1964). *Monte Carlo Methods*, Methuen & Co., London.
- Hemmerle, W.J. (1967). *Statistical Computations on a Digital Computer*. Blaisdell, Waltham, Massachusetts.
- Lange, K. (2004). Computational Statistics and Optimization Theory at UCLA, *The American Statistician*, 58:9–11.
- Milton, R. and Nelder, J. (eds) (1969). *Statistical Computation*, Academic Press, New York.
- Monahan, J. (2004). Teaching Statistical Computing at NC State, *The American Statistician*, 58:6–8.
- Von Neumann, J. (1951). *Various Techniques Used in Connection with Random Digits*, National Bureau of Standards Symposium, NBS Applied Mathematics Series 12, National Bureau of Standards (now National Institute of Standards and Technology), Washington, DC.
- Wegman, E.J. (1993). History of the Interface since 1987: The corporate era, *Computing Science and Statistics*, 25:27–32.
- Wilkinson, J. H. (1963). *Rounding Errors in Algebraic Processes*, Prentice-Hall, Inc., Englewood Cliffs, New Jersey.

Part II
Statistical Computing



Basic Computational Algorithms

II.1

John Monahan

1.1	<i>Computer Arithmetic</i>	20
	Integer Arithmetic	20
	Floating Point Arithmetic	21
	Cancellation	24
	Accumulated Roundoff Error	27
	Interval Arithmetic.....	27
1.2	<i>Algorithms</i>	27
	Iterative Algorithms	30
	Iterative Algorithms for Optimization and Nonlinear Equations	31

1.1

Computer Arithmetic

Numbers are the lifeblood of statistics, and computational statistics relies heavily on how numbers are represented and manipulated on a computer. Computer hardware and statistical software handle numbers well, and the methodology of computer arithmetic is rarely a concern. However, whenever we push hardware and software to their limits with difficult problems, we can see signs of the mechanics of floating point arithmetic around the frayed edges. To work on difficult problems with confidence and explore the frontiers of statistical methods and software, we need to have a sound understanding of the foundations of computer arithmetic. We need to know how arithmetic works and why things are designed the way they are.

As scientific computation began to rely heavily on computers, a monumental decision was made during the 1960's to change from base ten arithmetic to base two. Humans had been doing base ten arithmetic for only a few hundred years, during which time great advances were possible in science in a short period of time. Consequently, the resistance to this change was strong and understandable. The motivation behind the change to base two arithmetic is merely that it is so very easy to do addition (and subtraction) and multiplication in base two arithmetic. The steps are easy enough that a machine can be designed – wire a board of relays – or design a silicon chip – to do base two arithmetic. Base ten arithmetic is comparatively quite difficult, as its recent mathematical creation would suggest. However two big problems arise in changing from base ten to base two: (1) we need to constantly convert numbers written in base ten by humans to base two number system and then back again to base ten for humans to read the results, and (2) we need to understand the limits of arithmetic in a different number system.

1.1.1

Integer Arithmetic

Computers use two basic ways of writing numbers: fixed point (for integers) and floating point (for real numbers). Numbers are written on a computer following base two positional notation. The *positional number system* is a convention for expressing a number as a list of integers (digits), representing a number x in base B by a list of digits $a_m, a_{m-1}, \dots, a_1, a_0$ whose mathematical meaning is

$$x = a_{m-1}B^{m-1} + \dots + a_2B^2 + a_1B + a_0 \quad (1.1)$$

where the digits a_j are integers in $\{0, \dots, B - 1\}$. We are accustomed to what is known in the West as the Arabic numbers, $0, 1, 2, \dots, 9$ representing those digits for writing for humans to read. For base two arithmetic, only two digits are needed $\{0, 1\}$. For base sixteen, although often viewed as just a collection of four binary digits (1 *byte* = 4 *bits*), the Arabic numbers are augmented with letters, as $\{0, 1, 2, \dots, 9, a, b, c, d, e, f\}$, so that $f_{\text{sixteen}} = 15_{\text{ten}}$.

The system based on (1.1), known as *fixed point arithmetic*, is useful for writing integers. The choice of $m = 32$ dominates current computer hardware, although

smaller ($m = 16$) choices are available via software and larger ($m = 48$) hardware had been common in high performance computing. Recent advances in computer architecture may soon lead to the standard changing to $m = 64$. While the writing of a number in base two requires only the listing of its binary digits, a convention is necessary for expression of negative numbers. The survivor of many years of intellectual competition is the *two's complement* convention. Here the first (leftmost) bit is reserved for the sign, using the convention that 0 means positive and 1 means negative. Negative numbers are written by complementing each bit (replace 1 with 0, 0 with 1) and adding one to the result. For $m = 16$ (easier to display), this means that 22_{ten} and its negative are written as

$$(0\ 001\ 0110) = 22_{\text{ten}}$$

and

$$(1\ 110\ 1010) = -22_{\text{ten}} .$$

Following the two's complement convention with m bits, the smallest (negative) number that can be written is -2^{m-1} and the largest positive number is $2^{m-1} - 1$; zero has a unique representation of $(0\ 000\ \dots\ 0000)$. Basic arithmetic (addition and multiplication) using two's complement is easy to code, essentially taking the form of mod 2^{m-1} arithmetic, with special tools for overflow and sign changes. See, for example, Knuth (1997) for history and details, as well as algorithms for base conversions.

The great advantage of fixed point (integer) arithmetic is that it is so very fast. For many applications, integer arithmetic suffices, and most nonscientific computer software only uses fixed point arithmetic. Its second advantage is that it does not suffer from the rounding error inherent in its competitor, floating point arithmetic, whose discussion follows.

Floating Point Arithmetic

 1.1.2

To handle a larger subset of the real numbers, the positional notation system includes an exponent to express the location of the radix point (generalization of the decimal point), so that the usual format is a triple (sign, exponent, fraction) to represent a number as

$$x = (-1)^{\text{sign}} B^{\text{exponent}} (a_1 B^{-1} + a_2 B^{-2} + \dots + a_d B^{-d}) , \quad (1.2)$$

where the fraction is expressed by its list of base B digits $0.a_1 a_2 a_3 \dots a_d$. To preserve as much information as possible with the limited d digits to represent the fraction, *normalization* is usually enforced, that is, the leading/most significant digit a_1 is nonzero – except for the special case $x = 0$. The mathematical curiosity of an infinite series expansion of a number has no place here where only d digits are available. Moreover, a critical issue is what to do when only d digits are available. *Rounding* to the nearest number is preferred to the alternative *chopping*; in the case of representing $\pi = 3.14159265 \dots$ to $d = 5$ decimal ($B = \text{ten}$) digits leads to the more accurate $(+, +1, 0.31416)$ in the case of rounding, rather than $(+, +1,$

0.31415) for the chopping alternative. Notice that normalization and the use of this positional notation reflects a goal of preserving relative accuracy, or reducing the relative error in the approximation. The expression of a real number x in floating point arithmetic can be expressed mathematically in terms of a function $fl : R \rightarrow \mathcal{F}$ where \mathcal{F} is the set of numbers that can be represented using this notation, the set of floating point numbers. The relative accuracy of this rounding operation can be expressed as

$$fl(x) = (1 + u)x, \quad (1.3)$$

where $|u| \leq U$ where U is known as the *machine unit*. Seen in terms of the relative error of $fl(x)$ in approximating x , the expression above can be rewritten as

$$|x - fl(x)|/|x| \leq U \quad \text{for } x \neq 0.$$

For base B arithmetic with d digits and chopping, $U = B^{1-d}$; rounding reduces U by a factor of 2.

An important conceptual leap is the understanding that most numbers are represented only approximately in floating point arithmetic. This extends beyond the usual irrational numbers such as π or e that cannot be represented with a finite number of digits. A novice user may enter a familiar assignment such as $x = 8.6$ and, observing that the computer prints out 8.6000004, may consider this an error. When the “8.6” was entered, the computer had to first parse the text “8.6” and recognize the decimal point and arabic numbers as a representation, for humans, of a real number with the value $8 + 6 \times 10^{-1}$. The second step is to convert this real number to a base two floating point number – approximating this base ten number with the closest base two number – this is the function $fl(\cdot)$. Just as $1/3$ produces the repeating decimal 0.33333... in base 10, the number 8.6 produces a repeating binary representation 1000.100110011..._{two}, and is chopped or rounded to the nearest floating point number $fl(8.6)$. Later, in printing this same number out, a second conversion produces the closest base 10 number to $fl(8.6)$ with few digits; in this case 8.6000004, not an error at all. Common practice is to employ numbers that are integers divided by powers of two, since they are exactly represented. For example, distributing 1024 equally spaced points makes more sense than the usual 1000, since $j/1024$ can be exactly represented for any integer j .

A breakthrough in hardware for scientific computing came with the adoption and implementation of the IEEE 754 binary floating point arithmetic standard, which has standards for two levels of precision, *single precision* and *double precision* (IEEE, 1985). The single precision standard uses 32 bits to represent a number: a single bit for the sign, 8 bits for the exponent and 23 bits for the fraction. The double precision standard requires 64 bits, using 3 more bits for the exponent and adds 29 to the fraction for a total of 52. Since the leading digit of a normalized number is nonzero, in base two the leading digit must be one. As a result, the floating point form (1.2) above takes a slightly modified form:

$$x = (-1)^{\text{sign}} B^{\text{exponent} - \text{excess}} (1 + a_1 B^{-1} + a_2 B^{-2} + \dots + a_d B^{-d}) \quad (1.4)$$

as the fraction is expressed by its list of binary digits $1.a_1a_2a_3 \dots a_d$. As a result, while only 23 bits are stored, it works as if one more bit were stored. The exponent using 8 bits can range from 0 to 255; however, using an excess of 127, the range of the difference (*exponent* – *excess*) goes from –126 to 127. The finite number of bits available for storing numbers means that the set of floating point numbers \mathcal{F} is a finite, discrete set. Although well-ordered, it does have a largest number, smallest number, and smallest positive number. As a result, this IEEE Standard expresses positive numbers from approximately 1.4×10^{-45} to 3.4×10^{38} with a machine unit $U = 2^{-24} \approx 10^{-7}$ using only 31 bits. The remaining 32nd bit is reserved for the sign. Double precision expands the range to roughly $10^{\pm 300}$ with $U = 2^{-53} \approx 10^{-16}$, so the number of accurate digits is more than doubled.

The two extreme values of the exponent are employed for special features. At the high end, the case exponent = 255 signals two infinities ($\pm\infty$) with the largest possible fraction. These values arise as the result of an *overflow* operation. The most common causes are adding or multiplying two very large numbers, or from a function call that produces a result that is larger than any floating point number. For example, the value of $\exp(x)$ is larger than any finite number in \mathcal{F} for $x > 88.73$ in single precision. Before adoption of the standard, $\exp(89.9)$ would cause the program to cease operation due to this “exception”. Including $\pm\infty$ as members of \mathcal{F} permits the computations to continue, since a sensible result is now available. As a result, further computations involving the value $\pm\infty$ can proceed naturally, such as $1/\infty = 0$. Again using the exponent = 255, but with any other fraction represents *not-a-number*, usually written as “NaN”, and used to express the result of *invalid operations*, such as $0/0$, $\infty - \infty$, $0 \times \infty$, and square roots of negative numbers. For statistical purposes, another important use of NaN is to designate missing values in data. The use of infinities and NaN permit continued execution in the case of anomalous arithmetic operations, instead of causing computation to cease when such anomalies occur. The other extreme exponent = 0 signals a denormalized number with the net exponent of –126 and an unnormalized fraction, with the representation following (1.2), rather than the usual (1.4) with the unstated and unstored 1. The *denormalized* numbers further expand the available numbers in \mathcal{F} , and permit a *soft underflow*. Underflow, in contrast to overflow, arises when the result of an arithmetic operation is smaller in magnitude than the smallest representable positive number, usually caused by multiplying two small numbers together. These denormalized numbers begin approximately 10^{-38} near the reciprocal of the largest positive number. The denormalized numbers provide even smaller numbers, down to 10^{-45} . Below that, the next number in \mathcal{F} is the floating point zero: the smallest exponent and zero fraction – all bits zero.

Most statistical software employs only double precision arithmetic, and some users become familiar with apparent aberrant behavior such as a sum of residuals of 10^{-16} instead of zero. While many areas of science function quite well using single precision, some problems, especially nonlinear optimization, nevertheless require double precision. The use of single precision requires a sound understand of rounding error. However, the same rounding effects remain in double precision,

but because their effects are so often obscured from view, double precision may promote a naive view that computers are perfectly accurate.

The machine unit expresses a relative accuracy in storing a real number as a floating point number. Another similar quantity, the machine epsilon, denoted by ϵ_m , is defined as the smallest positive number than, when added to one, gives a result that is different from one. Mathematically, this can be written as

$$fl(1 + x) = 1 \text{ for } 0 < x < \epsilon_m . \quad (1.5)$$

Due to the limited precision in floating point arithmetic, adding a number that is much smaller in magnitude than the machine epsilon will not change the result. For example, in single precision, the closest floating point number to $1 + 2^{-26}$ is 1. Typically, both the machine unit and machine epsilon are nearly the same size, and these terms often used interchangeably without grave consequences.

1.1.3 Cancellation

Often one of the more surprising aspects of floating point arithmetic is that some of the more familiar laws of algebra are occasionally violated: in particular, the associative and distributive laws. While most occurrences are just disconcerting to those unfamiliar to computer arithmetic, one serious concern is cancellation. For a simple example, consider the case of base ten arithmetic with $d = 6$ digits, and take $x = 123.456$ and $y = 123.332$, and note that both x and y may have been rounded, perhaps x was 123.456478 or 123.456000 or 123.455998. Now x would be stored as $(+, 3, .123456)$ and y would be written as $(+, 3, .123332)$, and when these two numbers are subtracted, we have the unnormalized difference $(+, 3, .000124)$. Normalization would lead to $(+, 0, .124???)$ where merely “?” represents that some digits need to take their place. The simplistic option is to put zeros, but .124478 is just as good an estimate of the true difference between x and y as .124000, or .123998, for that matter. The problem with cancellation is that the relative accuracy that floating point arithmetic aims to protect has been corrupted by the loss of the leading significant digits. Instead of a small error in the sixth digit, we now have that error in the third digit; the relative error has effectively been magnified by a factor of 1000 due to the cancellation of the first 3 digits.

The best way to deal with the potential problem caused by catastrophic cancellation is to avoid them. In many cases, the cancellation may be avoided by reworking the computations analytically to handle the cancellation:

$$1 - (1 - 2t)^{-1} = \frac{1 - 2t - 1}{1 - 2t} = \frac{-2t}{1 - 2t} .$$

In this case, there is significant cancellation when t is small, and catastrophic cancellation whenever t drops below the machine epsilon. Using six digit decimal arithmetic to illustrate, at $t = 0.001$, the left hand expression, $1 - (1 - 2t)^{-1}$, gives

$$1.00000 - 1.00200 = .200000 \times 10^{-2}$$

while the right hand expression, $-2t/(1 - 2t)$, gives

$$.200401 \times 10^{-2} .$$

The relative error in using the left hand expression is an unacceptable .002. At $t = 10^{-7}$, the first expression leads to a complete cancellation yielding zero and a relative error of one. Just a little algebra here avoids the most of the effect of cancellation. When the expressions involve functions, cases where cancellation occurs can often be handled by approximations. In the case of $1 - e^{-t}$, serious cancellation will occur whenever t is very small. The cancellation can be avoided for this case by using a power series expansion:

$$1 - e^{-t} = 1 - (1 - t + t^2/2 - \dots) \approx t - t^2/2 = t(1 - t/2) .$$

When $t = .0001$, the expression $1 - e^{-t}$ leads to the steps

$$1.00000 - 0.999900 = .100000 \times 10^{-4} ,$$

while the approximation gives

$$(.0001)(.999950) = .999950 \times 10^{-4}$$

which properly approximates the result to six decimal digits. At $t = 10^{-5}$ and 10^{-6} , similar results occur, with complete cancellation at 10^{-7} . Often the approximation will be accurate just when cancellation must be avoided.

One application where rounding error must be understood and cancellation cannot be avoided is numerical differentiation, where calls to a function are used to approximate a derivative from a first difference:

$$f'(x) \approx [f(x+h) - f(x)]/h . \quad (1.6)$$

Mathematically, the accuracy of this approximation is improved by taking h very small; following a quadratic Taylor's approximation, we can estimate the error as

$$[f(x+h) - f(x)]/h \approx f'(x) + \frac{1}{2}hf''(x) .$$

However, when the function calls $f(x)$ and $f(x+h)$ are available only to limited precision – a relative error of ϵ_m , taking h smaller leads to more cancellation. The cancellation appears as a random rounding error in the numerator of (1.6) which becomes magnified by dividing by a small h . Taking h larger incurs more bias from the approximation; taking h smaller incurs larger variance from the rounding error. Prudence dictates balancing bias and variance. Dennis and Schnabel (1983) recommend using $h \approx \epsilon_m^{1/2}$ for first differences, but see also Bodily (2002).

The second approach for avoiding the effects of cancellation is to develop different methods. A common cancellation problem in statistics arises from using the formula

$$\sum_{i=1}^n y_i^2 - n\bar{y}^2 \quad (1.7)$$

for computing the sum of squares around the mean. Cancellation can be avoided by following the more familiar two-pass method

$$\sum_{i=1}^n (y_i - \bar{y})^2 \quad (1.8)$$

but this algorithm requires all of the observations to be stored and repeated updates are difficult. A simple adjustment to avoid cancellation, requiring only a single pass and little storage, uses the first observation to center:

$$\sum_{i=1}^n (y_i - \bar{y})^2 = \sum_{i=1}^n (y_i - y_1)^2 - n(y_1 - \bar{y})^2 . \quad (1.9)$$

An orthogonalization method from regression using Givens rotations (see Chan et al., 1983) can do even better:

$$t_i = t_{i-1} + y_i \quad (1.10)$$

$$s_i = s_{i-1} + (iy_i - t_i)^2 / (i(i-1)) . \quad (1.11)$$

To illustrate the effect of cancellation, take the simple problem of $n = 5$ observations, $y_i = 4152 + i$ so that $y_1 = 4153$ through $y_5 = 4157$. Again using six decimal digits, the computations of the sum and mean encounter no problems, and we easily get $\bar{y} = 4155$ or $.415500 \times 10^4$, and $\sum y_i = 20775$ or $.207750 \times 10^5$. However, each square loses some precision in rounding:

$$\begin{aligned} y_1 = 4153 , \quad y_1^2 = 4153^2 = 17,247,409 \quad \text{rounded to} \quad .172474 \times 10^8 \\ y_2 = 4154 , \quad y_2^2 = 4154^2 = 17,255,716 \quad \text{rounded to} \quad .172557 \times 10^8 \\ y_3 = 4155 , \quad y_3^2 = 4155^2 = 17,264,025 \quad \text{rounded to} \quad .172640 \times 10^8 \\ y_4 = 4156 , \quad y_4^2 = 4156^2 = 17,272,336 \quad \text{rounded to} \quad .172723 \times 10^8 \\ y_5 = 4157 , \quad y_5^2 = 4157^2 = 17,280,649 \quad \text{rounded to} \quad .172806 \times 10^8 . \end{aligned}$$

Summing the squares encounters no further rounding on its way to 0.863200×10^8 , and we compute the corrected sum of squares as

$$\begin{aligned} .863200 \times 10^8 - (.207750 \times 10^5) \times 4155 \\ .863200 \times 10^8 - .863201 \times 10^8 = -100 . \end{aligned}$$

The other three algorithms, following (1.8), (1.9), (1.10), and (1.11), each give the perfect result of 28 in this case.

Admittedly, while this example is contrived to show an absurd result, a negative sum of squares, the equally absurd value of zero is hardly unusual. Similar computations – differences of sum of squares – are routine, especially in regression and

in the computation of eigenvalues and eigenvectors. In regression, the orthogonalization method (1.10) and (1.11) is more commonly seen in its general form. In all these cases, simply centering can improve the computational difficulty and reduce the effect of limited precision arithmetic.

Accumulated Roundoff Error

1.1.4

Another problem with floating point arithmetic is the sheer accumulation of rounding error. While many applications run well in spite of a large number of calculations, some approaches encounter surprising problems. An enlightening example is just to add up many ones: $1 + 1 + 1 + \dots$. Astonishingly, this infinite series appears to converge – the partial sums stop increasing as soon as the ratio of the new number to be added, in this case, one, to the current sum (n) drops below the machine epsilon. Following (1.5), we have $fl(n + 1) = fl(n)$, from which we find

$$1/n \approx \varepsilon_m \quad \text{or} \quad n \approx 1/\varepsilon_m .$$

So you will find the infinite series of ones converging to $1/\varepsilon_m$. Moving to double precision arithmetic pushes this limit of accuracy sufficiently far to avoid most problems – but it does not eliminate them. A good mnemonic for assessing the effect of accumulated rounding error is that doing m additions amplifies the rounding error by a factor of m . For single precision, adding 1000 numbers would look like a relative error of 10^{-4} which is often unacceptable, while moving to double precision would lead to an error of 10^{-13} . Avoidance strategies, such as adding smallest to largest and nested partial sums, are discussed in detail in Monahan, (2001, Chap. 2).

Interval Arithmetic

1.1.5

One of the more interesting methods for dealing with the inaccuracies of floating point arithmetic is interval arithmetic. The key is that a computer can only do arithmetic operations: addition, subtraction, multiplication, and division. The novel idea, though, is that instead of storing the number x , its lower and upper bounds (\underline{x}, \bar{x}) are stored, designating an interval for x . Bounds for each of these arithmetic operations can be then established as functions of the input. For addition, the relationship can be written as:

$$\underline{x} + \underline{y} < x + y < \bar{x} + \bar{y} .$$

Similar bounds for the other three operations can be established. The propagation of rounding error through each step is then captured by successive upper and lower bounds on intermediate quantities. This is especially effective in probability calculations using series or continued fraction expansions. The final result is an interval that we can confidently claim contains the desired calculation. The hope is always that interval is small. Software for performing interval arithmetic has been implemented in a practical fashion by modifying a Fortran compiler. See, for

example, Hayes (2003) for an introductory survey, and Kearfott and Kreinovich (1996) for articles on applications.

1.2 Algorithms

An algorithm is a list of directed actions to accomplish a designated task. Cooking recipes are the best examples of algorithms in everyday life. The level of a cookbook reflect the skill of the cook: a gourmet cookbook may include the instruction “saute the onion until transparent” while a beginner’s cookbook would describe how to choose and slice the onion, what kind of pan, the level of heat, etc. Since computers are inanimate objects incapable of thought, instructions for a computer algorithm must go much, much further to be completely clear and unambiguous, and include all details.

Most cooking recipes would be called *single pass* algorithms, since they are a list of commands to be completed in consecutive order. Repeating the execution of the same tasks, as in baking batches of cookies, would be described in algorithmic terms as *looping*. Looping is the most common feature in mathematical algorithms, where a specific task, or similar tasks, are to be repeated many times. The computation of an inner product is commonly implemented using a loop:

$$\mathbf{a}^T \mathbf{b} = a_1 b_1 + a_2 b_2 + \dots + a_n b_n,$$

implemented as

```
s = 0
do i = 1 to n
    s = s + ai × bi
end do
```

where the range of the loop includes the single statement with a multiplication and addition. In an *iterative* algorithm, the number of times the loop is be repeated is not known in advance, but determined by some monitoring mechanism. For mathematical algorithms, the focus is most often monitoring convergence of a sequence or series. Care must be taken in implementing iterative algorithms to insure that, at some point, the loop will be terminated, otherwise an improperly coded procedure may proceed indefinitely in an *infinite loop*. Surprises occur when the convergence of an algorithm can be proven analytically, but, because of the discrete nature of floating point arithmetic, the procedure implementing that algorithm may not converge. For example, in a square-root problem to be examined further momentarily, we cannot find $x \in \mathcal{F}$ so that $x \times x$ is exactly equal to 2. The square of one number may be just below two, and the square of the next largest number in \mathcal{F} may be larger than 2. When monitoring convergence, common practice is to convert any test for equality of two floating point numbers or expressions to tests of closeness:

$$\text{if } (\text{abs}(x^*x - 2) < \text{eps}) \text{ then exit.} \quad (1.12)$$

Most mathematical algorithms have more sophisticated features. Some algorithms are *recursive*, employing relationships such as the gamma function: $\Gamma(x+1) = x\Gamma(x)$ so that new values can be computed using previous values. Powerful recursive algorithms, such as the Fast Fourier Transform (FFT) and sorting algorithms, follow a *divide-and-conquer* paradigm: to solve a big problem, break it into little problems and use the solutions to the little problems to solve the big problem. In the case of sorting, the algorithm may look something like:

```

algorithm sort(list)
break list into two pieces: first and second
sort (first)
sort (second)
put sorted lists first and second together to form
  one sorted list
end algorithm sort

```

Implemented recursively, a big problem is quickly broken into tiny pieces and the key to the performance of divide-and-conquer algorithms is in combining the solutions to lots of little problems to address the big problem. In cases where these solutions can be easily combined, these recursive algorithms can achieve remarkable breakthroughs in performance. In the case of sorting, the standard algorithm, known as bubblesort, takes $O(n^2)$ work to sort a problem of size n – if the size of the problem is doubled, the work goes up by factor of 4. The Discrete Fourier Transform, when written as the multiplication of an $n \times n$ matrix and a vector, involves n^2 multiplications and additions. In both cases, the problem is broken into two subproblems, and the mathematics of divide and conquer follows a simple recursive relationship, that the time/work $T(n)$ to solve a problem of size n is the twice the time/work to solve two subproblem with half the size, plus the time/work $C(n)$, to put the solutions together:

$$T(n) = 2T(n/2) + C(n). \quad (1.13)$$

In both sorting and the Discrete Fourier Transform, $C(n) \approx cn + d$, which leads to $T(n) = cn \log(n) + O(n)$. A function growing at the rate $O(n \log n)$ grows so much slower than $O(n^2)$, that the moniker “Fast” in Fast Fourier Transform is well deserved. While some computer languages preclude the use of recursion, recursive algorithms can often be implemented without explicit recursion through clever programming.

The performance of an algorithm may be measured in many ways, depending on the characteristics of the problems the it may be intended to solve. The sample variance problem above provides an example. The simple algorithm using (1.7) requires minimal storage and computation, but may lose accuracy when the variance is much smaller than the mean: the common test problem for exhibiting catastrophic cancellation employs $y_i = 2^{12} + i$ for single precision. The two-pass method (1.8) requires all of the observations to be stored, but provides the most accuracy and least computation. Centering using the first observation (1.9) is near-

ly as fast, requires no extra storage, and its accuracy only suffers when the first observation is unlike the others. The last method, arising from the use of Givens transformations (1.10) and (1.11), also requires no extra storage, gives sound accuracy, but requires more computation. As commonly seen in the marketplace of ideas, the inferior methods have not survived, and the remaining competitors all have tradeoffs with speed, storage, and numerical stability.

1.2.1 Iterative Algorithms

The most common difficult numerical problems in statistics involve optimization, or root-finding: maximum likelihood, nonlinear least squares, M-estimation, solving the likelihood equations or generalized estimating equations. And the algorithms for solving these problems are typically iterative algorithms, using the results from the current step to direct the next step.

To illustrate, consider the problem of computing the square root of a real number y . Following from the previous discussion of floating point arithmetic, we can restrict y to the interval $(1, 2)$. One approach is to view the problem as a root-finding problem, that is, we seek x such that $f(x) = x^2 - y = 0$. The bisection algorithm is a simple, stable method for finding a root. In this case, we may start with an interval known to contain the root, say (x_1, x_2) , with $x_1 = 1$ and $x_2 = 2$. Then bisection tries $x_3 = 1.5$, the midpoint of the current interval. If $f(x_3) < 0$, then $x_3 < \sqrt{y} < x_2$, and the root is known to belong in the new interval (x_3, x_2) . The algorithm continues by testing the midpoint of the current interval, and eliminating half of the interval. The rate of convergence of this algorithm is linear, since the interval of uncertainty, in this case, is cut by a constant $(1/2)$ with each step. For other algorithms, we may measure the rate at which the distance from the root decreases. Adapting Newton's method to this root-finding problem yields Heron's iteration

$$x_{n+1} = \frac{1}{2}(x_n + y/x_n) .$$

Denoting the solution as $x^* = \sqrt{y}$, the error at step n can be defined as $\varepsilon_n = x_n - x^*$, leading to the relationship

$$\varepsilon_{n+1} = \frac{1}{2} \frac{\varepsilon_n^2}{x_n} . \quad (1.14)$$

This relationship of the errors is usually called *quadratic convergence*, since the new error is proportional to the square of the error at the previous step. The relative error $\delta_n = (x_n - x^*)/x^*$ follows a similar relationship,

$$\delta_n = \frac{1}{2} \delta_n^2 / (1 + \delta_n) . \quad (1.15)$$

Here, the number of accurate digits is doubled with each iteration. For the secant algorithm, analysis of the error often leads to a relationship similar to (1.14), but $|\varepsilon_{n+1}| \approx C|\varepsilon_n|^p$, with $1 < p < 2$, achieving a rate of convergence known as

superlinear. For some well-defined problems, as the square root problem above, the number of iterations needed to reduce the error or relative error below some criterion can be determined in advance.

While we can stop this algorithm when $f(x_n) = 0$, as discussed previously, there may not be any floating point number that will give a zero to the function, hence the stopping rule (1.12). Often in root-finding problems, we stop when $|f(x_n)|$ is small enough. In some problems, the appropriate “small enough” quantity to ensure the desired accuracy may depend on parameters of the problem, as in this case, the value of γ . As a result, termination criterion for the algorithm is changed to: stop when the relative change in x is small

$$|x_{n+1} - x_n|/|x_n| < \delta .$$

While this condition may cause premature stopping in rare cases, it will prevent infinite looping in other cases. Many optimization algorithms permit the iteration to be terminated using any combination – and “small enough” is within the user’s control. Nevertheless, unless the user learns a lot about the nature of the problem at hand, an unrealistic demand for accuracy can lead to unachievable termination criteria, and an endless search.

As discussed previously, rounding error with floating point computation affects the level of accuracy that is possible with iterative algorithms for root-finding. In general, the relative error in the root is at the same relative level as the computation of the function. While optimization problems have many of the same characteristics as root-finding problems, the effect of computational error is a bit more substantial: k digits of accuracy in the function to be optimization can produce but $k/2$ digits in the root/solution.

Iterative Algorithms for Optimization and Nonlinear Equations

1.2.2

In the multidimensional case, the common problems are solving a system of nonlinear equations or optimizing a function of several variables. The most common tools for these problems are Newton’s method or secant-like variations. Given the appropriate regularity conditions, again we can achieve quadratic convergence with Newton’s method, and superlinear convergence with secant-like variations. In the case of optimization, we seek to minimize $f(x)$, and Newton’s method is based on minimizing the quadratic approximation:

$$f(\mathbf{x}) \approx f(\mathbf{x}_0) + (\mathbf{x} - \mathbf{x}_0)^\top \nabla f(\mathbf{x}_0) + (\mathbf{x} - \mathbf{x}_0)^\top \nabla^2 f(\mathbf{x}_0)(\mathbf{x} - \mathbf{x}_0) .$$

This leads to the iteration step

$$\mathbf{x}^{(n+1)} = \mathbf{x}^{(n)} - [\nabla^2 f(\mathbf{x}^{(n)})]^{-1} \nabla f(\mathbf{x}^{(n)}) .$$

In the case of solving a system of nonlinear equations, $\mathbf{g}(\mathbf{x}) = \mathbf{0}$, Newton’s method arises from solving the affine (linear) approximation

$$\mathbf{g}(\mathbf{x}) \approx \mathbf{g}(\mathbf{x}_0) + \mathbf{J}_g(\mathbf{x}_0)(\mathbf{x} - \mathbf{x}_0) ,$$

leading to a similar iteration step

$$\mathbf{x}^{(n+1)} = \mathbf{x}^{(n)} - [J_g(\mathbf{x}^{(n)})]^{-1} \mathbf{g}(\mathbf{x}^{(n)}) .$$

In both cases, under suitable smoothness conditions, the Newton iteration will achieve quadratic convergence – using norms to measure the error at each step:

$$\|\mathbf{x}^{(n+1)} - \mathbf{x}^*\| \approx C \|\mathbf{x}^{(n)} - \mathbf{x}^*\|^2 .$$

For both problems, Newton’s method requires the computation of lots of derivatives, either the gradient $\nabla f(\mathbf{x}_0)$ and Hessian $\nabla^2 f(\mathbf{x}_0)$, or the Jacobian matrix $J_g(\mathbf{x}^{(n)})$. In the univariate root-finding problem, the secant method arises by approximating the derivative with the first difference using the previous evaluation of the function. Secant analogues can be constructed for both the optimization and nonlinear equations problems, with similar reduction in the convergence rate: from quadratic to superlinear.

In both problems, the scaling of the parameters is quite important, as measuring the error with the Euclidean norm presupposes that errors in each component are equally weighted. Most software for optimization includes a parameter vector for suitably scaling the parameters, so that one larger parameter does not dominate the convergence decision. In solving nonlinear equations, the condition of the problem is given by

$$\|J_g(\mathbf{x}^{(n)})\| \|[J_g(\mathbf{x}^{(n)})]^{-1}\|$$

(as in solving linear equations) and scaling problem extends to the components of $\mathbf{g}(\mathbf{x})$. In many statistical problems, such as robust regression, the normal parameter scaling issues arise with the covariates and their coefficients. However, one component of $\mathbf{g}(\mathbf{x})$, associated with the error scale parameter may be orders of magnitude larger or smaller than the other equations. As with parameter scaling, this is often best done by the user and is not easily overcome automatically.

With the optimization problem, there is a natural scaling with $\nabla f(\mathbf{x}_0)$ in contrast with the Jacobian matrix. Here, the eigenvectors of the Hessian matrix $\nabla^2 f(\mathbf{x}_0)$ dictate the condition of the problem; see, for example, Gill et al. (1981) and Dennis and Schnabel (1983). Again, parameter scaling remains one of the most important tools.

References

- Bodily, C.H. (2002). *Numerical Differentiation Using Statistical Design*. Ph.D. Thesis, NC State University.
- Chan, T.F., Golub, G.H. and LeVeque, R.J. (1983). Algorithms for computing the sample variance, *American Statistician*, 37:242–7.
- Dennis, J.E. Jr. and Schnabel, R.B. (1983). *Numerical Methods for Unconstrained Optimization*. Englewood Cliffs, NJ, Prentice-Hall.

- Gill, P.E., Murray, W. and Wright, M.H. (1981). *Practical Optimisation*, London, Academic Press.
- Hayes, B. (2003). A lucid interval, *American Scientist*, 91:484–488.
- Institute of Electrical and Electronics Engineers (1985). *A Proposed IEEE-CS Standard for Binary Floating Point Arithmetic*, Standard, 754–1985, IEEE, New York.
- Kearfott, R.B. and Kreinovich, V. (eds) (1996). *Applications of Interval Computations*, Boston, Kluwer.
- Knuth, D.E. (1997). *The Art of Computer Programming*, (Vol. 2: Seminumerical Algorithms), Third Edition, Reading MA, Addison-Wesley.
- Monahan, J.F. (2001). *Numerical Methods of Statistics*. Cambridge University Press.
- Overton, M.L. (2001). *Numerical Computing with IEEE Floating Point Arithmetic*. Philadelphia, SIAM.

Random Number Generation

II.2

Pierre L'Ecuyer

2.1	<i>Introduction</i>	36
2.2	<i>Uniform Random Number Generators</i>	36
	Physical Devices	37
	Generators Based on a Deterministic Recurrence	37
	Quality Criteria	38
	Statistical Testing.....	40
	Cryptographically Strong Generators	41
2.3	<i>Linear Recurrences Modulo m</i>	41
	The Multiple Recursive Generator	41
	The Lattice Structure.....	42
	MRG Implementation Techniques	45
	Combined MRGs and LCGs	46
	Jumping Ahead.....	47
	Linear Recurrences With Carry	48
2.4	<i>Generators Based on Recurrences Modulo 2</i>	49
	A General Framework	49
	Measures of Uniformity	50
	Lattice Structure in Spaces of Polynomials and Formal Series	51
	The LFSR Generator	52
	The GFSR and Twisted GFSR	52
	Combined Linear Generators Over \mathbb{F}_2	53
2.5	<i>Nonlinear RNGs</i>	54
2.6	<i>Examples of Statistical Tests</i>	55
2.7	<i>Available Software and Recommendations</i>	57
2.8	<i>Non-uniform Random Variate Generation</i>	57
	Inversion.....	58

The Alias Method	60
Kernel Density Estimation and Generation.....	61
The Rejection Method.....	61
Thinning for Point Processes with Time-varying Rates	62
The Ratio-of-Uniforms Method	62
Composition and Convolution	63
Other Special Techniques.....	63

Introduction

The fields of probability and statistics are built over the abstract concepts of probability space and random variable. This has given rise to elegant and powerful mathematical theory, but exact implementation of these concepts on conventional computers seems impossible. In practice, random variables and other random objects are *simulated* by *deterministic algorithms*. The purpose of these algorithms is to produce sequences of numbers or objects whose behavior is very hard to distinguish from that of their “truly random” counterparts, at least for the application of interest. Key requirements may differ depending on the context. For Monte Carlo methods, the main goal is to reproduce the statistical properties on which these methods are based, so that the Monte Carlo estimators behave as expected, whereas for gambling machines and cryptology, observing the sequence of output values for some time should provide no practical advantage for predicting the forthcoming numbers better than by just guessing at random.

In computational statistics, random variate generation is usually made in two steps: (1) generating imitations of independent and identically distributed (i.i.d.) random variables having the uniform distribution over the interval $(0, 1)$ and (2) applying transformations to these i.i.d. $U(0, 1)$ random variates in order to generate (or imitate) random variates and random vectors from arbitrary distributions. These two steps are essentially independent and the world’s best experts on them are two different groups of scientists, with little overlap. The expression (*pseudo*)*random number generator* (RNG) usually refers to an algorithm used for Step (1).

In principle, the simplest way of generating a random variate X with distribution function F from a $U(0, 1)$ random variate U is to apply the inverse of F to U :

$$X = F^{-1}(U) \stackrel{\text{def}}{=} \min\{x \mid F(x) \geq U\}. \quad (2.1)$$

This is the *inversion* method. It is easily seen that X has the desired distribution: $P[X \leq x] = P[F^{-1}(U) \leq x] = P[U \leq F(x)] = F(x)$. Other methods are sometimes preferable when F^{-1} is too difficult or expensive to compute, as will be seen later.

The remainder of this chapter is organized as follows. In the next section, we give a definition and the main requirements of a uniform RNG. Generators based on linear recurrences modulo a large integer m , their lattice structure and quality criteria, and their implementation, are covered in Sect. 2.3. In Sect. 2.4, we have a similar discussion for RNGs based on linear recurrences modulo 2. Nonlinear RNGs are briefly presented in Sect. 2.5. In Sect. 2.6, we discuss empirical statistical testing of RNGs and give some examples. Section 2.7 contains a few pointers to recommended RNGs and software. In Sect. 2.8, we cover non-uniform random variate generators. We first discuss inversion and its implementation in various settings. We then explain the alias, rejection, ratio-of-uniform, composition, and convolution methods, and provide pointers to the several other methods that apply in special cases.

Important basic references that we recommend are Knuth (1998), L'Ecuyer (1994, 1998), Niederreiter (1992), and Tezuka (1995) for uniform RNGs, and Devroye (1986), Gentle (2003), and Hörmann et al. (2004) for non-uniform RNGs.

2.2 Uniform Random Number Generators

2.2.1 Physical Devices

Random numbers can be generated via physical mechanisms such as the timing between successive events in atomic decay, thermal noise in semiconductors, and the like. A key issue when constructing a RNG based on a physical device is that a “random” or “chaotic” output does not suffice; the numbers produced must be, at least to a good approximation, realizations of *independent and uniformly distributed* random variables. If the device generates a stream of bits, which is typical, then each bit should be 0 or 1 with equal probability, and be independent of all the other bits. In general, this cannot be proved, so one must rely on the results of empirical statistical testing to get convinced that the output values have the desired statistical behavior. Not all these devices are reliable, but some apparently are. I did test two of them recently and they passed all statistical tests that I tried.

For computational statistics, physical devices have several disadvantages compared to a good algorithmic RNG that stands in a few lines of code. For example, (a) they are much more cumbersome to install and run; (b) they are more costly; (c) they are slower; (d) they cannot reproduce exactly the same sequence twice. Item (d) is important in several contexts, including program verification and debugging as well as comparison of similar systems by simulation with common random numbers to reduce the variance (Bratley et al., 1987; Fishman, 1996; Law and Kelton, 2000). Nevertheless, these physical RNGs can be useful for selecting the seed of an algorithmic RNG, more particularly for applications in cryptology and for gaming machines, where frequent reseeding of the RNG with an external source of entropy (or randomness) is important. A good algorithmic RNG whose seed is selected at random can be viewed as an extensor of randomness, stretching a short random seed into a long sequence of *pseudorandom* numbers.

2.2.2 Generators Based on a Deterministic Recurrence

RNGs used for simulation and other statistical applications are almost always based on deterministic algorithms that fit the following framework, taken from L'Ecuyer (1994): a RNG is a structure $(\mathcal{S}, \mu, f, \mathcal{U}, g)$ where \mathcal{S} is a finite set of *states* (the *state space*), μ is a probability distribution on \mathcal{S} used to select the *initial state* (or *seed*) s_0 , $f : \mathcal{S} \rightarrow \mathcal{S}$ is the *transition function*, \mathcal{U} is the *output space*, and $g : \mathcal{S} \rightarrow \mathcal{U}$ is the *output function*. Usually, $\mathcal{U} = (0, 1)$, and we shall assume

henceforth that this is the case. The state of the RNG evolves according to the recurrence $s_i = f(s_{i-1})$, for $i \geq 1$, and the *output* at step i is $u_i = g(s_i) \in \mathcal{U}$. The output values u_0, u_1, u_2, \dots are the so-called *random numbers* produced by the RNG.

Because \mathcal{S} is finite, there must be some finite $l \geq 0$ and $j > 0$ such that $s_{l+j} = s_l$. Then, for all $i \geq l$, one has $s_{i+j} = s_i$ and $u_{i+j} = u_i$, because both f and g are deterministic. That is, the state and output sequences are eventually periodic. The smallest positive j for which this happens is called the *period length* of the RNG, and is denoted by ρ . When $l = 0$, the sequence is said to be *purely periodic*. Obviously, $\rho \leq |\mathcal{S}|$, the cardinality of \mathcal{S} . If the state has a k -bit representation on the computer, then $\rho \leq 2^k$. Good RNGs are designed so that their period length ρ is not far from that upper bound. In general, the value of ρ may depend on the seed s_0 , but good RNGs are normally designed so that the period length is the same for all admissible seeds.

In practical implementations, it is important that the output be *strictly* between 0 and 1, because $F^{-1}(U)$ is often infinite when U is 0 or 1. All good implementations take care of that. However, for the mathematical analysis of RNGs, we often assume that the output space is $[0, 1)$ (i.e., 0 is admissible), because this simplifies the analysis considerably without making much difference in the mathematical structure of the generator.

Quality Criteria

2.2.3

What important quality criteria should we consider when designing RNGs? An extremely *long period* is obviously essential, to make sure that no wrap-around over the cycle can occur in practice. The length of the period must be guaranteed by a mathematical proof. The RNG must also be *efficient* (run fast and use only a small amount of memory), *repeatable* (able to reproduce exactly the same sequence as many times as we want), and *portable* (work the same way in different software/hardware environments). The availability of efficient *jump-ahead* methods that can quickly compute $s_{i+\nu}$ given s_i , for any large ν and any i , is also very useful, because it permits one to partition the RNG sequence into long disjoint *streams* and *substreams* of random numbers, in order to create an arbitrary number of *virtual generators* from a single RNG (Law and Kelton, 2000; L'Ecuyer et al., 2002a). These virtual generators can be used on parallel processors or to support different sources of randomness in a large simulation model, for example.

Consider a RNG with state space $\mathcal{S} = \{1, \dots, 2^{1000} - 1\}$, transition function $s_{i+1} = f(s_i) = (s_i + 1) \bmod 2^{1000}$, and $u_i = g(s_i) = s_i/2^{1000}$. This RNG has period length 2^{1000} and enjoys all the nice properties described in the preceding paragraph, but is far from imitating “randomness”. In other words, these properties are *not sufficient*.

A sequence of real-valued random variables u_0, u_1, u_2, \dots are i.i.d. $U(0, 1)$ if and only if for every integers $i \geq 0$ and $t > 0$, the vector $\mathbf{u}_{i,t} = (u_i, \dots, u_{i+t-1})$ is uniformly distributed over the t -dimensional unit hypercube $(0, 1)^t$. Of course,

this cannot hold for algorithmic RNGs because any vector of t successive values produced by the generator must belong to the *finite set*

$$\Psi_t = \{(u_0, \dots, u_{t-1}) : s_0 \in \mathcal{S}\},$$

which is the set of all vectors of t successive output values, from all possible initial states. Here we interpret Ψ_t as a *multiset*, which means that the vectors are counted as many times as they appear, and the cardinality of Ψ_t is exactly equal to that of \mathcal{S} .

Suppose we select the seed s_0 at random, uniformly over \mathcal{S} . This can be approximated by using some physical device, for example. Then, the vector $\mathbf{u}_{0,t}$ has the uniform distribution over the finite set Ψ_t . And if the sequence is purely periodic for all s_0 , $\mathbf{u}_{i,t} = (u_i, \dots, u_{i+t-1})$ is also uniformly distributed over Ψ_t for all $i \geq 0$. Since the goal is to approximate the uniform distribution over $(0, 1)^t$, it immediately becomes apparent that Ψ_t should be evenly spread over this unit hypercube. In other words, Ψ_t *approximates* $(0, 1)^t$ as the *sample space* from which the vectors of successive output values are drawn randomly, so it must be a good approximation of $(0, 1)^t$ in some sense. The design of good-quality RNGs must therefore involve practical ways of measuring the uniformity of the corresponding sets Ψ_t even when they have huge cardinalities. In fact, a large state space \mathcal{S} is necessary to obtain a long period, but an even more important reason for having a huge number of states is to make sure that Ψ_t can be large enough to provide a good uniform coverage of the unit hypercube, at least for moderate values of t .

More generally, we may also want to measure the uniformity of sets of the form

$$\Psi_I = \{(u_{i_1}, \dots, u_{i_t}) \mid s_0 \in \mathcal{S}\},$$

where $I = \{i_1, \dots, i_t\}$ is a fixed set of non-negative integers such that $0 \leq i_1 < \dots < i_t$. As a special case, we recover $\Psi_t = \Psi_I$ when $I = \{0, \dots, t-1\}$.

The uniformity of a set Ψ_I is typically assessed by measuring the *discrepancy* between the empirical distribution of its points and the uniform distribution over $(0, 1)^t$ (Niederreiter, 1992; Hellekalek and Larcher, 1998; L'Ecuyer and Lemieux, 2002). Discrepancy measures are equivalent to goodness-of-fit test statistics for the multivariate uniform distribution. They can be defined in many different ways. In fact, the choice of a specific definition typically depends on the mathematical structure of the RNG to be studied and the reason for this is very pragmatic: we must be able to compute these measures quickly even when \mathcal{S} has very large cardinality. This obviously excludes any method that requires explicit generation of the sequence over its entire period. The selected discrepancy measure is usually computed for each set I in some predefined class \mathcal{J} , these values are weighted or normalized by factors that depend on I , and the worst-case (or average) over \mathcal{J} is adopted as a *figure of merit* used to rank RNGs. The choice of \mathcal{J} and of the weights are arbitrary. Typically, \mathcal{J} would contain sets I such that t and $i_t - i_1$ are rather small. Examples of such figures of merit will be given when we discuss specific classes of RNGs.

Statistical Testing

2.2.4

Good RNGs are designed based on mathematical analysis of their properties, then implemented and submitted to batteries of *empirical statistical tests*. These tests try to detect empirical evidence against the null hypothesis \mathcal{H}_0 : “the u_i are realizations of i.i.d. $U(0, 1)$ random variables”. A test can be defined by any function T that maps a sequence u_0, u_1, \dots in $(0, 1)$ to a real number X , and for which a good approximation is available for the distribution of the random variable X under \mathcal{H}_0 . For the test to be implementable, X must depend on only a finite (but perhaps random) number of u_i 's. Passing many tests may improve one's confidence in the RNG, but never guarantees that the RNG is foolproof for all kinds of simulations.

Building a RNG that *passes all statistical tests* is an impossible dream. Consider, for example, the class of all tests that examine the first (most significant) b bits of n successive output values, u_0, \dots, u_{n-1} , and return a binary value $X \in \{0, 1\}$. Select $\alpha \in (0, 1)$ so that αb^n is an integer and let $\mathcal{T}_{n,b,\alpha}$ be the tests in this class that return $X = 1$ for *exactly* αb^n of the b^n possible output sequences. We may say that the sequence *fails* the test when $X = 1$. The number of tests in $\mathcal{T}_{n,b,\alpha}$ is equal to the number of ways of choosing αb^n distinct objects among b^n . The chosen objects are the sequences that fail the test. Now, for any given output sequence, the number of such tests that return 1 for this particular sequence is equal to the number of ways of choosing the other $\alpha b^n - 1$ sequences that also fail the test. This is the number of ways of choosing $\alpha b^n - 1$ distinct objects among $b^n - 1$. In other words, as pointed out by Leeb (1995), every output sequence fails exactly the same number of tests! This result should not be surprising. Viewed from a different angle, it is essentially a restatement of the well-known fact that under \mathcal{H}_0 , each of the b^n possible sequences has the same probability of occurring, so one could argue that none should be considered more random than any other (Knuth, 1998).

This viewpoint seems to lead into a dead end. For statistical testing to be meaningful, all tests should not be considered on equal footing. So which ones are more important? Any answer is certainly tainted with its share of arbitrariness. However, for large values of n , the number of tests is huge and all but a tiny fraction are too complicated even to be implemented. So we may say that *bad* RNGs are those that fail simple tests, whereas *good* RNGs fail only complicated tests that are hard to find and run. This common-sense compromise has been generally adopted in one way or another.

Experience shows that RNGs with very long periods, good structure of their set Ψ_t , and based on recurrences that are not too simplistic, pass most reasonable tests, whereas RNGs with short periods or bad structures are usually easy to crack by standard statistical tests. For sensitive applications, it is a good idea, when this is possible, to apply additional statistical tests designed in close relation with the random variable of interest (e.g., based on a *simplification* of the stochastic model being simulated, and for which the theoretical distribution can be computed).

Our discussion of statistical tests continues in Sect. 2.6.

2.2.5 Cryptographically Strong Generators

One way of defining an ideal RNG would be that no statistical test can distinguish its output sequence from an i.i.d. $U(0, 1)$ sequence. If an unlimited computing time is available, no finite-state RNG can satisfy this requirement, because by running it long enough one can eventually figure out its periodicity. But what if we impose a limit on the computing time? This can be analyzed formally in the framework of asymptotic *computational complexity* theory, under the familiar “rough-cut” assumption that polynomial-time algorithms are practical and others are not.

Consider a family of RNGs $\{\mathcal{G}_k = (\mathcal{S}_k, \mu_k, f_k, \mathcal{U}_k, g_k), k = 1, 2, \dots\}$ where \mathcal{S}_k is of cardinality 2^k (i.e., \mathcal{G}_k has a k -bit state). Suppose that the transition and output functions f and g can be computed in time bounded by a polynomial in k . Let \mathcal{T} be the class of statistical tests that run in time bounded by a polynomial in k and try to differentiate between the output sequence of the RNG and an i.i.d. $U(0, 1)$ sequence. The RNG family is called *polynomial-time perfect* if there is a constant $\varepsilon > 0$ such that for all k , no test in \mathcal{T} can differentiate correctly with probability larger than $1/2 + e^{-k\varepsilon}$. This is equivalent to asking that no polynomial-time algorithm can predict any given bit of u_i with probability of success larger than $1/2 + e^{-k\varepsilon}$, after observing u_0, \dots, u_{i-1} . This links unpredictability with statistical uniformity and independence. For the proofs and additional details, see, e.g. Blum et al. (1986), L'Ecuyer and Proulx (1989), Lagarias (1993), and Luby (1996). This theoretical framework has been used to define a notion of reliable RNG in the context of cryptography. But the guarantee is only asymptotic; it does not necessarily tell what value of k is large enough for the RNG to be secure in practice. Moreover, specific RNG families have been proved to be polynomial-time perfect only under yet unproven conjectures. So far, no one has been able to prove even their existence. Most RNGs discussed in the remainder of this chapter are known *not* to be polynomial-time perfect. However, they are fast, convenient, and have good enough statistical properties when their parameters are chosen carefully.

2.3 Linear Recurrences Modulo m

2.3.1 The Multiple Recursive Generator

The most widely used RNGs are based on the linear recurrence

$$x_i = (a_1 x_{i-1} + \dots + a_k x_{i-k}) \bmod m, \quad (2.2)$$

where m and k are positive integers called the *modulus* and the *order*, and the *coefficients* a_1, \dots, a_k are in \mathbb{Z}_m , interpreted as the set $\{0, \dots, m-1\}$ on which all operations are performed with reduction modulo m . The *state* at step i is $s_i = \mathbf{x}_i = (x_{i-k+1}, \dots, x_i)^\top$. When m is a prime number, the finite ring \mathbb{Z}_m is a finite field and it is possible to choose the coefficients a_j so that the period length

reaches $\rho = m^k - 1$ (the largest possible value) (Knuth, 1998). This maximal period length is achieved if and only if the characteristic polynomial of the recurrence, $P(z) = z^k - a_1 z^{k-1} - \dots - a_k$, is a primitive polynomial over \mathbb{Z}_m , i.e., if and only if the smallest positive integer ν such that $(z^\nu \bmod P(z)) \bmod m = 1$ is $\nu = m^k - 1$. Knuth (1998) explains how to verify this for a given $P(z)$. For $k > 1$, for $P(z)$ to be a primitive polynomial, it is necessary that a_k and at least another coefficient a_j be nonzero. Finding primitive polynomials of this form is generally easy and they yield the simplified recurrence:

$$x_n = (a_r x_{n-r} + a_k x_{n-k}) \bmod m. \tag{2.3}$$

A *multiple recursive generator* (MRG) uses (2.2) with a large value of m and defines the output as $u_i = x_i/m$. For $k = 1$, this is the classical *linear congruential generator* (LCG). In practice, the output function is modified slightly to make sure that u_i never takes the value 0 or 1 (e.g., one may define $u_i = (x_i + 1)/(m + 1)$, or $u_i = x_i/(m + 1)$ if $x_i > 0$ and $u_i = m/(m + 1)$ otherwise) but to simplify the theoretical analysis, we will follow the common convention of assuming that $u_i = x_i/m$ (in which case u_i does take the value 0 occasionally).

The Lattice Structure

Let e_i denote the i th unit vector in k dimensions, with a 1 in position i and 0's elsewhere. Denote by $x_{i,0}, x_{i,1}, x_{i,2}, \dots$ the values of x_0, x_1, x_2, \dots produced by the recurrence (2.2) when the initial state \mathbf{x}_0 is e_i . An arbitrary initial state $\mathbf{x}_0 = (z_1, \dots, z_k)^\top$ can be written as the linear combination $z_1 e_1 + \dots + z_k e_k$ and the corresponding sequence is a linear combination of the sequences $(x_{i,0}, x_{i,1}, \dots)$, with reduction of the coordinates modulo m . Reciprocally, any such linear combination reduced modulo m is a sequence that can be obtained from some initial state $\mathbf{x}_0 \in \mathcal{S} = \mathbb{Z}_m^k$. If we divide everything by m we find that for the MRG, for each $t \geq 1$, $\Psi_t = L_t \cap [0, 1)^t$ where

$$L_t = \left\{ \mathbf{v} = \sum_{i=1}^t z_i \mathbf{v}_i \mid z_i \in \mathbb{Z} \right\},$$

is a t -dimensional *lattice* in \mathbb{R}^t , with basis

$$\begin{aligned} \mathbf{v}_1 &= (1, 0, \dots, 0, x_{1,k}, \dots, x_{1,t-1})^\top / m \\ &\vdots \\ \mathbf{v}_k &= (0, 0, \dots, 1, x_{k,k}, \dots, x_{k,t-1})^\top / m \\ \mathbf{v}_{k+1} &= (0, 0, \dots, 0, 1, \dots, 0)^\top \\ &\vdots \\ \mathbf{v}_t &= (0, 0, \dots, 0, 0, \dots, 1)^\top. \end{aligned}$$

For $t \leq k$, L_t contains all vectors whose coordinates are multiples of $1/m$. For $t > k$, it contains a fraction m^{k-t} of those vectors.

This lattice structure implies that the points of Ψ_t are distributed according to a very regular pattern, in equidistant parallel hyperplanes. Graphical illustrations of this, usually for LCGs, can be found in a myriad of papers and books; e.g., Gentle (2003), Knuth (1998), Law and Kelton (2000), and L'Ecuyer (1998). Define the *dual lattice* to L_t as

$$L_t^* = \{\mathbf{h} \in \mathbb{R}^t : \mathbf{h}^\top \mathbf{v} \in \mathbb{Z} \text{ for all } \mathbf{v} \in L_t\}.$$

Each $\mathbf{h} \in L_t^*$ is a normal vector that defines a family of equidistant parallel hyperplanes, at distance $1/\|\mathbf{h}\|_2$ apart, and these hyperplanes cover all the points of L_t unless \mathbf{h} is an integer multiple of some other vector $\mathbf{h}' \in L_t^*$. Therefore, if ℓ_t is the euclidean length of a shortest non-zero vector \mathbf{h} in L_t^* , then there is a family of hyperplanes at distance $1/\ell_t$ apart that cover all the points of L_t . A small ℓ_t means thick slices of empty space between the hyperplanes and we want to avoid that. A large ℓ_t means a better (more uniform) coverage of the unit hypercube by the point set Ψ_t . Computing the value of $1/\ell_t$ is often called the *spectral test* (Knuth, 1998; Fishman, 1996).

The lattice property holds as well for the point sets Ψ_I formed by values at arbitrary lags defined by a fixed set of indices $I = \{i_1, \dots, i_t\}$. One has $\Psi_I = L_I \cap [0, 1)^t$ for some lattice L_I , and the largest distance between successive hyperplanes for a family of hyperplanes that cover all the points of L_I is $1/\ell_I$, where ℓ_I is the euclidean length of a shortest nonzero vector in L_I^* , the dual lattice to L_I .

The lattice L_I and its dual can be constructed as explained in Couture and L'Ecuyer (1996) and L'Ecuyer and Couture (1997). Finding the shortest nonzero vector in a lattice with basis $\mathbf{v}_1, \dots, \mathbf{v}_t$ can be formulated as an integer programming problem with a quadratic objective function:

$$\text{Minimize } \|\mathbf{v}\|^2 = \sum_{i=1}^t \sum_{j=1}^t z_i \mathbf{v}_i^\top \mathbf{v}_j z_j$$

subject to z_1, \dots, z_t integers and not all zero. This problem can be solved by a branch-and-bound algorithm (Fincke and Pohst, 1985; L'Ecuyer and Couture, 1997; Tezuka, 1995).

For any given dimension t and m^k points per unit of volume, there is an absolute upper bound on the best possible value of ℓ_I (Conway and Sloane, 1999; Knuth, 1998; L'Ecuyer, 1999b). Let $\ell_I^*(m^k)$ denote such an upper bound. To define a figure of merit that takes into account several sets I , in different numbers of dimensions, it is common practice to divide ℓ_I by an upper bound, in order to obtain a standardized value between 0 and 1, and then take the worst case over a given class \mathcal{J} of sets I . This gives a figure of merit of the form

$$M_{\mathcal{J}} = \min_{I \in \mathcal{J}} \ell_I / \ell_{|I|}^*(m^k).$$

A value of $M_{\mathcal{J}}$ too close to zero means that L_I has a bad lattice structure for at least one of the selected sets I . We want a value as close to 1 as possible. Computer searches for good MRGs with respect to this criterion have been reported by L'Ecuyer et al. (1993), L'Ecuyer and Andres (1997), L'Ecuyer (1999a), for example. In most cases, \mathcal{J} was simply the sets of the form $I = \{1, \dots, t\}$ for $t \leq t_1$, where t_1 was an arbitrary integer ranging from 8 to 45. L'Ecuyer and Lemieux (2000) also consider the small dimensional sets I with indices not too far apart. They suggest taking $\mathcal{J} = \{\{0, 1, \dots, i\} : i < t_1\} \cup \{\{i_1, i_2\} : 0 = i_1 < i_2 < t_2\} \cup \dots \cup \{\{i_1, \dots, i_d\} : 0 = i_1 < \dots < i_d < t_d\}$ for some positive integers d, t_1, \dots, t_d . We could also take a weighted average instead of the minimum in the definition of $M_{\mathcal{J}}$.

An important observation is that for $t > k$, the t -dimensional vector $\mathbf{h} = (-1, a_1, \dots, a_k, 0, \dots, 0)^T$ always belongs to L_t^* , because for any vector $\mathbf{v} \in L_t$, the first $k + 1$ coordinates of $m\mathbf{v}$ must satisfy the recurrence (2.2), which implies that $(-1, a_1, \dots, a_k, 0, \dots, 0)\mathbf{v}$ must be an integer. Therefore, one always has $\ell_t^2 \leq 1 + a_1^2 + \dots + a_k^2$. Likewise, if I contains 0 and all indices j such that $a_{k-j} \neq 0$, then $\ell_I^2 \leq 1 + a_1^2 + \dots + a_k^2$ (L'Ecuyer, 1997). This means that the sum of squares of the coefficients a_j must be large if we want to have any chance that the lattice structure be good.

Constructing MRGs with only two nonzero coefficients and taking these coefficients small has been a very popular idea, because this makes the implementation easier and faster (Deng and Lin, 2000; Knuth, 1998). However, MRGs thus obtained have a bad structure. As a worst-case illustration, consider the widely-available additive or subtractive *lagged-Fibonacci* generator, based on the recurrence (2.2) where the two coefficients a_r and a_k are both equal to ± 1 . In this case, whenever I contains $\{0, k - r, k\}$, one has $\ell_I^2 \leq 3$, so the distance between the hyperplanes is at least $1/\sqrt{3}$. In particular, for $I = \{0, k - r, k\}$, all the points of Ψ_I (aside from the zero vector) are contained in only two planes! This type of structure can have a dramatic effect on certain simulation problems and is a good reason for staying away from these lagged-Fibonacci generators, regardless of their parameters.

A similar problem occurs for the "fast MRG" proposed by Deng and Lin (2000), based on the recurrence

$$x_i = (-x_{i-1} + a x_{i-k}) \bmod m = ((m - 1)x_{i-1} + a x_{i-k}) \bmod m ,$$

with $a^2 < m$. If a is small, the bound $\ell_I^2 \leq 1 + a^2$ implies a bad lattice structure for $I = \{0, k - 1, k\}$. A more detailed analysis by L'Ecuyer and Touzin (2004) shows that this type of generator cannot have a good lattice structure even if the condition $a^2 < m$ is removed. Another special case proposed by Deng and Xu (2003) has the form

$$x_i = a (x_{i-j_2} + \dots + x_{i-j_t}) \bmod m . \tag{2.4}$$

In this case, for $I = \{0, k - j_{t-1}, \dots, k - j_2, k\}$, the vectors $(1, a, \dots, a)$ and $(a^*, 1, \dots, 1)$ both belong to the dual lattice L_I^* , where a^* is the multiplicative inverse of $a \bmod m$. So neither a nor a^* should be small.

To get around this structural problem when I contains certain sets of indices, Lüscher (1994) and Knuth (1998) recommend to skip some of the output values in order to break up the bad vectors. For the lagged-Fibonacci generator, for example, one can output k successive values produced by the recurrence, then skip the next d values, output the next k , skip the next d , and so on. A large value of d (e.g., $d = 5k$ or more) may get rid of the bad structure, but slows down the generator. See Wegenkittl and Matsumoto (1999) for further discussion.

2.3.3 MRG Implementation Techniques

The modulus m is often taken as a large prime number close to the largest integer directly representable on the computer (e.g., equal or near $2^{31} - 1$ for 32-bit computers). Since each x_{i-j} can be as large as $m - 1$, one must be careful in computing the right side of (2.2) because the product $a_j x_{i-j}$ is typically not representable as an ordinary integer. Various techniques for computing this product modulo m are discussed and compared by Fishman (1996), L'Ecuyer and Côté (1991), L'Ecuyer (1999a), and L'Ecuyer and Simard (1999). Note that if $a_j = m - a'_j > 0$, using a_j is equivalent to using the negative coefficient $-a'_j$, which is sometimes more convenient from the implementation viewpoint. In what follows, we assume that a_j can be either positive or negative.

One approach is to perform the arithmetic modulo m in 64-bit (double precision) floating-point arithmetic (L'Ecuyer, 1999a). Under this representation, assuming that the usual IEEE floating-point standard is respected, all positive integers up to 2^{53} are represented exactly. Then, if each coefficient a_j is selected to satisfy $|a_j|(m - 1) \leq 2^{53}$, the product $|a_j|x_{i-j}$ will always be represented exactly and $z_j = |a_j|x_{i-j} \bmod m$ can be computed by the instructions

$$y = |a_j|x_{i-j}; \quad z_j = y - m \lfloor y/m \rfloor.$$

Similarly, if $(|a_1| + \dots + |a_k|)(m - 1) \leq 2^{53}$, $a_1 x_{i-1} + \dots + a_k x_{i-k}$ will always be represented exactly.

A second technique, called *approximate factoring* (L'Ecuyer and Côté, 1991), uses only the integer representation and works under the condition that $|a_j| = i$ or $|a_j| = \lfloor m/i \rfloor$ for some integer $i < \sqrt{m}$. One precomputes $q_j = \lfloor m/|a_j| \rfloor$ and $r_j = m \bmod |a_j|$. Then, $z_j = |a_j|x_{i-j} \bmod m$ can be computed by

$$y = \lfloor x_{i-j}/q_j \rfloor; \quad z = |a_j|(x_{i-j} - yq_j) - yr_j;$$

if $z < 0$ then $z_j = z + m$ else $z_j = z$.

All quantities involved in these computations are integers between $-m$ and m , so no overflow can occur if m can be represented as an ordinary integer (e.g., $m < 2^{31}$ on a 32-bit computer).

The *powers-of-two decomposition* approach selects coefficients a_j that can be written as a sum or difference of a small number of powers of 2 (Wu, 1997; L'Ecuyer and Simard, 1999; L'Ecuyer and Touzin, 2000). For example, one may take

$a_j = \pm 2^q \pm 2^r$ and $m = 2^e - h$ for some positive integers $q, r, e,$ and h . To compute $y = 2^q x \bmod m$, decompose $x = z_0 + 2^{e-q} z_1$ (where $z_0 = x \bmod 2^{e-q}$) and observe that

$$y = 2^q(z_0 + 2^{e-q}z_1) \bmod(2^e - h) = (2^qz_0 + hz_1) \bmod(2^e - h) .$$

Suppose now that

$$h < 2^q \quad \text{and} \quad h(2^q - (h + 1)2^{-e+q}) < m . \tag{2.5}$$

Then, $2^q z_0 < m$ and $h z_1 < m$, so y can be computed by shifts, masks, additions, subtractions, and a single multiplication by h . Intermediate results never exceed $2m - 1$. Things simplify further if $q = 0$ or $q = 1$ or $h = 1$. For $h = 1$, y is obtained simply by swapping the blocks of bits z_0 and z_1 (Wu, 1997). It has been pointed out by L'Ecuyer and Simard (1999) that LCGs with parameters of the form $m = 2^e - 1$ and $a = \pm 2^q \pm 2^r$ have bad statistical properties because the recurrence does not “mix the bits” well enough. However, good and fast MRGs can be obtained via the power-of-two decomposition method, as explained in L'Ecuyer and Touzin (2000).

Another interesting idea for improving efficiency is to take all nonzero coefficients a_j equal to the same constant a (Marsaglia, 1996; Deng and Xu, 2003). Then, computing the right side of (2.2) requires a single multiplication. Deng and Xu (2003) provide specific parameter sets and concrete implementations for MRGs of this type, for prime m near 2^{31} , and $k = 102, 120,$ and 1511 .

One may be tempted to take m equal to a power of two, say $m = 2^e$, because then the “mod m ” operation is much easier: it suffices to keep the e least significant bits and mask-out all others. However, taking a power-of-two modulus is not recommended because it has several strong disadvantages in terms of the quality of the RNG (L'Ecuyer, 1990,1998). In particular, the least significant bits have very short periodicity and the period length of the recurrence (2.2) cannot exceed $(2^k - 1)2^{e-1}$ if $k > 1$, and 2^{e-2} if $k = 1$ and $e \geq 4$. The maximal period length achievable with $k = 7$ and $m = 2^{31}$, for example, is more than 2^{180} times smaller than the maximal period length achievable with $k = 7$ and $m = 2^{31} - 1$ (a prime number).

Combined MRGs and LCGs

The conditions that make MRG implementations run faster (e.g., only two nonzero coefficients both close to zero) are generally in conflict with those required for having a good lattice structure and statistical robustness. *Combined MRGs* are one solution to this problem. Consider J distinct MRGs evolving in parallel, based on the recurrences

$$x_{j,i} = (a_{j,1}x_{j,i-1} + \cdots + a_{j,k}x_{j,i-k}) \bmod m_j ; \tag{2.6}$$

where $a_{j,k} \neq 0$, for $j = 1, \dots, J$. Let $\delta_1, \dots, \delta_J$ be arbitrary integers,

$$z_i = (\delta_1 x_{1,i} + \dots + \delta_J x_{J,i}) \bmod m_1, \quad u_i = z_i / m_1, \quad (2.7)$$

and

$$w_i = (\delta_1 x_{1,i} / m_1 + \dots + \delta_J x_{J,i} / m_J) \bmod 1. \quad (2.8)$$

This defines two RNGs, with output sequences $\{u_i, i \geq 0\}$ and $\{w_i, i \geq 0\}$.

Suppose that the m_j are pairwise relatively prime, that δ_j and m_j have no common factor for each j , and that each recurrence (2.6) is purely periodic with period length φ_j . Let $m = m_1 \cdots m_J$ and let ϱ be the least common multiple of $\varphi_1, \dots, \varphi_J$. Under these conditions, the following results have been proved by L'Ecuyer and Tezuka (1991) and L'Ecuyer (1996a): (a) the sequence (2.8) is exactly equivalent to the output sequence of a MRG with (composite) modulus m and coefficients a_j that can be computed explicitly as explained in L'Ecuyer (1996a); (b) the two sequences in (2.7) and (2.8) have period length ϱ ; and (c) if both sequences have the same initial state, then $u_i = w_i + \varepsilon_i$ where $\max_{i \geq 0} |\varepsilon_i|$ can be bounded explicitly by a constant ε which is very small when the m_j are close to each other.

Thus, these combined MRGs can be viewed as practical ways of implementing an MRG with a large m and several large nonzero coefficients. The idea is to cleverly select the components so that: (1) each one is easy to implement efficiently (e.g., has only two small nonzero coefficients) and (2) the MRG that corresponds to the combination has a good lattice structure. If each m_j is prime and if each component j has maximal period length $\varphi_j = m_j^k - 1$, then each φ_j is even and ϱ cannot exceed $\varphi_1 \cdots \varphi_J / 2^{J-1}$. Tables of good parameters for combined MRGs of different sizes that reach this upper bound are given in L'Ecuyer (1999a) and L'Ecuyer and Touzin (2000), together with C implementations.

2.3.5 Jumping Ahead

The recurrence (2.2) can be written in matrix form as

$$\mathbf{x}_i = \mathbf{A} \mathbf{x}_{i-1} \bmod m = \begin{pmatrix} 0 & 1 & \cdots & 0 \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \\ a_k & a_{k-1} & \cdots & a_1 \end{pmatrix} \mathbf{x}_{i-1} \bmod m.$$

To jump ahead directly from \mathbf{x}_i to \mathbf{x}_{i+v} , for an arbitrary integer v , it suffices to exploit the relationship

$$\mathbf{x}_{i+v} = \mathbf{A}^v \mathbf{x}_i \bmod m = (\mathbf{A}^v \bmod m) \mathbf{x}_i \bmod m.$$

If this is to be done several times for the same ν , the matrix $A^\nu \bmod m$ can be precomputed once for all. For a large ν , this can be done in $O(\log_2 \nu)$ matrix multiplications via a standard divide-and-conquer algorithm (Knuth, 1998):

$$A^\nu \bmod m = \begin{cases} (A^{\nu/2} \bmod m)(A^{\nu/2} \bmod m) \bmod m & \text{if } \nu \text{ is even;} \\ A(A^{\nu-1} \bmod m) \bmod m & \text{if } \nu \text{ is odd.} \end{cases}$$

Linear Recurrences With Carry

2.3.6

These types of recurrences were introduced by Marsaglia and Zaman (1991) to obtain a large period even when m is a power of two (in which case the implementation may be faster). They were studied and generalized by Tezuka et al. (1994), Couture and L'Ecuyer (1994), Couture and L'Ecuyer (1997), and Goresky and Klapper (2003). The basic idea is to add a *carry* to the linear recurrence (2.2). The general form of this RNG, called *multiply-with-carry* (MWC), can be written as

$$x_i = (a_1 x_{i-1} + \dots + a_k x_{i-k} + c_{i-1})d \bmod b, \quad (2.9)$$

$$c_i = \lfloor (a_0 x_i + a_1 x_{i-1} + \dots + a_k x_{i-k} + c_{i-1})/b \rfloor, \quad (2.10)$$

$$u_i = \sum_{\ell=1}^{\infty} x_{i+\ell-1} b^{-\ell}, \quad (2.11)$$

where b is a positive integer (e.g., a power of two), a_0, \dots, a_k are arbitrary integers such that a_0 is relatively prime to b , and d is the multiplicative inverse of $-a_0$ modulo b . The state at step i is $s_i = (x_{i-k+1}, \dots, x_i, c_i)^T$. In practice, the sum in (2.11) is truncated to a few terms (it could be a single term if b is large), but the theoretical analysis is much easier for the infinite sum.

Define $m = \sum_{\ell=0}^k a_\ell b^\ell$ and let a be the inverse of b in arithmetic modulo m , assuming for now that $m > 0$. A major result proved in Tezuka et al. (1994), Couture and L'Ecuyer (1997), and Goresky and Klapper (2003) is that if the initial states agree, the output sequence $\{u_i, i \geq 0\}$ is exactly the same as that produced by the LCG with modulus m and multiplier a . Therefore, the MWC can be seen as a clever way of implementing a LCG with very large modulus. It has been shown by Couture and L'Ecuyer (1997) that the value of ℓ_t for this LCG satisfies $\ell_t^2 \leq a_0^2 + \dots + a_k^2$ for $t \geq k$, which means that the lattice structure will be bad unless the sum of squares of coefficients a_j is large.

In the original proposals of Marsaglia and Zaman (1991), called *add-with-carry* and *subtract-with-borrow*, one has $-a_0 = \pm a_r = \pm a_k = 1$ for some $r < k$ and the other coefficients a_j are zero, so $\ell_t^2 \leq 3$ for $t \geq k$ and the generator has essentially the same structural defect as the additive lagged-Fibonacci generator. In the version studied by Couture and L'Ecuyer (1997), it was assumed that $-a_0 = d = 1$. Then, the period length cannot exceed $(m-1)/2$ if b is a power of two. A concrete

implementation was given in that paper. Goresky and Klapper (2003) pointed out that the maximal period length of $\varphi = m - 1$ can be achieved by allowing a more general a_0 . They provided specific parameters that give a maximal period for b ranging from 2^{21} to 2^{35} and φ up to approximately 2^{2521} .

Generators Based on Recurrences Modulo 2

2.4

2.4.1 A General Framework

It seems natural to exploit the fact that computers work in binary arithmetic and to design RNGs defined directly in terms of bit strings and sequences. We do this under the following framework, taken from L'Ecuyer and Panneton (2002). Let \mathbb{F}_2 denote the finite field with two elements, 0 and 1, in which the operations are equivalent to addition and multiplication modulo 2. Consider the RNG defined by a matrix linear recurrence over \mathbb{F}_2 , as follows:

$$\mathbf{x}_i = \mathbf{A}\mathbf{x}_{i-1}, \quad (2.12)$$

$$\mathbf{y}_i = \mathbf{B}\mathbf{x}_i, \quad (2.13)$$

$$u_i = \sum_{\ell=1}^w y_{i,\ell-1} 2^{-\ell} = y_{i,0} y_{i,1} y_{i,2} \cdots, \quad (2.14)$$

where $\mathbf{x}_i = (x_{i,0}, \dots, x_{i,k-1})^T \in \mathbb{F}_2^k$ is the k -bit *state vector* at step i , $\mathbf{y}_i = (y_{i,0}, \dots, y_{i,w-1})^T \in \mathbb{F}_2^w$ is the w -bit *output vector* at step i , k and w are positive integers, \mathbf{A} is a $k \times k$ *transition matrix* with elements in \mathbb{F}_2 , \mathbf{B} is a $w \times k$ *output transformation matrix* with elements in \mathbb{F}_2 , and $u_i \in [0, 1)$ is the *output* at step i . All operations in (2.12) and (2.13) are performed in \mathbb{F}_2 .

It is well-known (Niederreiter, 1992; L'Ecuyer, 1994) that when the \mathbf{x}_i 's obey (2.12), for each j , the sequence $\{x_{i,j}, i \geq 0\}$ follows the linear recurrence

$$x_{i,j} = (\alpha_1 x_{i-1,j} + \cdots + \alpha_k x_{i-k,j}) \bmod 2, \quad (2.15)$$

whose *characteristic polynomial* $P(z)$ is the characteristic polynomial of \mathbf{A} , i.e.,

$$P(z) = \det(\mathbf{A} - z\mathbf{I}) = z^k - \alpha_1 z^{k-1} - \cdots - \alpha_{k-1} z - \alpha_k,$$

where \mathbf{I} is the identity matrix and each α_j is in \mathbb{F}_2 . The sequences $\{y_{i,j}, i \geq 0\}$, for $0 \leq j < w$, also obey the same recurrence (although some of them may follow recurrences of shorter order as well in certain situations, depending on \mathbf{B}). We assume that $\alpha_k = 1$, so that the recurrence (2.15) has *order* k and is purely periodic. Its period length is $2^k - 1$ (i.e., maximal) if and only if $P(z)$ is a primitive polynomial over \mathbb{F}_2 (Niederreiter, 1992; Knuth, 1998).

To jump ahead directly from x_i to x_{i+v} with this type of generator, it suffices to precompute the matrix A^v (in \mathbb{F}_2) and then multiply x_i by this matrix.

Several popular classes of RNGs fit this framework as special cases, by appropriate choices of the matrices A and B . This includes the Tausworthe or LFSR, polynomial LCG, GFSR, twisted GFSR, Mersenne twister, multiple recursive matrix generators, and combinations of these (L'Ecuyer and Panneton, 2002; Matsumoto and Nishimura, 1998; Niederreiter, 1995; Tezuka, 1995). We detail some of them after discussing measures of their uniformity.

Measures of Uniformity

2.4.2

The uniformity of point sets Ψ_I produced by RNGs based on linear recurrences over \mathbb{F}_2 is usually assessed by measures of equidistribution defined as follows (L'Ecuyer, 1996b; L'Ecuyer and Panneton, 2002; L'Ecuyer, 2004; Tezuka, 1995). For an arbitrary vector $q = (q_1, \dots, q_t)$ of non-negative integers, partition the unit hypercube $[0, 1]^t$ into 2^{q_j} intervals of the same length along axis j , for each j . This determines a partition of $[0, 1]^t$ into $2^{q_1 + \dots + q_t}$ rectangular boxes of the same size and shape. We call this partition the q -equidissection of the unit hypercube.

For some index set $I = \{i_1, \dots, i_t\}$, if Ψ_I has 2^k points, we say that Ψ_I is q -equidistributed in base 2 if there are exactly 2^q points in each box of the q -equidissection, where $k - q = q_1 + \dots + q_t$. This means that among the 2^k points $(x_{j_1}, \dots, x_{j_t})$ of Ψ_I , if we consider the first q_1 bits of x_{j_1} , the first q_2 bits of x_{j_2}, \dots , and the first q_t bits of x_{j_t} , each of the 2^{k-q} possibilities occurs exactly the same number of times. This is possible only if $q \leq k$.

The q -equidistribution of Ψ_I depends only on the first q_j bits of x_{i_j} for $1 \leq j \leq t$, for the points $(x_{i_1}, \dots, x_{i_t})$ that belong to Ψ_I . The vector of these $q_1 + \dots + q_t = k - q$ bits can always be expressed as a linear function of the k bits of the initial state x_0 , i.e., as $M_q x_0$ for some $(k - q) \times k$ binary matrix M_q , and it is easily seen that Ψ_I is q -equidistributed if and only if M_q has full rank $k - q$. This provides an easy way of checking equidistribution (Fushimi, 1983; L'Ecuyer, 1996b; Tezuka, 1995).

If Ψ_I is (ℓ, \dots, ℓ) -equidistributed for some $\ell \geq 1$, it is called t -distributed with ℓ bits of accuracy, or (t, ℓ) -equidistributed (L'Ecuyer, 1996b). The largest value of ℓ for which this holds is called the resolution of the set Ψ_I and is denoted by ℓ_I . This value has the upper bound $\ell_t^* = \min(\lfloor k/t \rfloor, w)$. The resolution gap of Ψ_I is defined as $\delta_I = \ell_t^* - \ell_I$. In the same vein as for MRGs, a worst-case figure of merit can be defined here by

$$\Delta_{\mathcal{J}} = \max_{I \in \mathcal{J}} \delta_I,$$

where \mathcal{J} is a preselected class of index sets I .

The point set Ψ_I is a (q, k, t) -net in base 2 (often called a (t, m, s) -net in the context of quasi-Monte Carlo methods, where a different notation is used (Niederreiter, 1992)), if it is (q_1, \dots, q_t) -equidistributed in base 2 for all non-negative integers q_1, \dots, q_t summing to $k - q$. We call the smallest such q the q -value of Ψ_I .

The smaller it is, the better. One candidate for a figure of merit could be the q -value of Ψ_t for some large t . A major drawback of this measure is that it is extremely difficult to compute for good long-period generators (for which $k - q$ is large), because there are too many vectors \mathbf{q} for which equidistribution needs to be checked. In practice, one must settle for figures of merit that involve a smaller number of equidissections.

When $\delta_I = 0$ for all sets I of the form $I = \{0, \dots, t - 1\}$, for $1 \leq t \leq k$, the RNG is said to be *maximally equidistributed* or *asymptotically random* for the word size w (L'Ecuyer, 1996b; Tezuka, 1995; Tootill et al., 1973). This property ensures perfect equidistribution of all sets Ψ_t , for any partition of the unit hypercube into subcubes of equal sizes, as long as $\ell \leq w$ and the number of subcubes does not exceed the number of points in Ψ_t . Large-period maximally equidistributed generators, together with their implementations, can be found in L'Ecuyer (1999c), L'Ecuyer and Panneton (2002), and Panneton and L'Ecuyer (2004), for example.

2.4.3 Lattice Structure in Spaces of Polynomials and Formal Series

The RNGs defined via (2.12)–(2.14) do not have a lattice structure in the real space like MRGs, but they do have a lattice structure in a space of formal series, as explained in Couture and L'Ecuyer (2000), L'Ecuyer (2004), Lemieux and L'Ecuyer (2003), and Tezuka (1995). The real space \mathbb{R} is replaced by the space \mathbb{L}_2 of formal power series with coefficients in \mathbb{F}_2 , of the form $\sum_{\ell=\omega}^{\infty} x_\ell z^{-\ell}$ for some integer ω . In that setting, the lattices have the form

$$\mathcal{L}_t = \left\{ \mathbf{v}(z) = \sum_{j=1}^t h_j(z) \mathbf{v}_j(z) \quad \text{such that each } h_j(z) \in \mathbb{F}_2[z] \right\},$$

where $\mathbb{F}_2[z]$ is the ring of polynomials with coefficients in \mathbb{F}_2 , and the basis vectors $\mathbf{v}_j(z)$ are in \mathbb{L}_2^t . The elements of the dual lattice \mathcal{L}_t^* are the vectors $\mathbf{h}(z)$ in \mathbb{L}_2^t whose scalar product with any vector of \mathcal{L}_t is a polynomial (in $\mathbb{F}_2[z]$). We define the mapping $\varphi : \mathbb{L}_2 \rightarrow \mathbb{R}$ by

$$\varphi \left(\sum_{\ell=\omega}^{\infty} x_\ell z^{-\ell} \right) = \sum_{\ell=\omega}^{\infty} x_\ell 2^{-\ell}.$$

Then, it turns out that the point set Ψ_t produced by the generator is equal to $\varphi(\mathcal{L}_t) \cap [0, 1)^t$. Moreover, the equidistribution properties examined in Sect. 2.4.2 can be expressed in terms of lengths of shortest vectors in the dual lattice, with appropriate definitions of the length (or norm). Much of the theory and algorithms developed for lattices in the real space can be adapted to these new types of lattices (Couture and L'Ecuyer, 2000; L'Ecuyer, 2004; Lemieux and L'Ecuyer, 2003; Tezuka, 1995).

The LFSR Generator

2.4.4

The *Tausworthe* or *linear feedback shift register* (LFSR) generator (Tausworthe, 1965; L'Ecuyer, 1996b; Tezuka, 1995) is a special case of (2.12)–(2.14) with $A = A_0^s$ (in \mathbb{F}_2) for some positive integer s , where

$$A_0 = \begin{pmatrix} & & & 1 & & \\ & & & & \ddots & \\ & & & & & 1 \\ a_k & a_{k-1} & \dots & a_1 & & \end{pmatrix}, \quad (2.16)$$

a_1, \dots, a_k are in \mathbb{F}_2 , $a_k = 1$, and all blank entries in the matrix are zeros. We take $w \leq k$ and the matrix B contains the first w lines of the $k \times k$ identity matrix. The RNG thus obtained can be defined equivalently by

$$x_i = a_1 x_{i-1} + \dots + a_k x_{i-k} \pmod{2}, \quad (2.17)$$

$$u_i = \sum_{\ell=1}^w x_{is+\ell-1} 2^{-\ell}. \quad (2.18)$$

Here, $P(z)$ is the characteristic polynomial of the matrix A_0^s , not the characteristic polynomial of the recurrence (2.17), and the choice of s is important for determining the quality of the generator. A frequently encountered case is when a single a_j is nonzero in addition to a_k ; then, $P(z)$ is a trinomial and we say that we have a *trinomial-based* LFSR generator. These generators are known to have important statistical deficiencies (Matsumoto and Kurita, 1996; Tezuka, 1995) but they can be used as components of combined RNGs.

LFSR generators can be expressed as LCGs in a space of polynomials (Tezuka and L'Ecuyer, 1991; Tezuka, 1995; L'Ecuyer, 1994). With this representation, their lattice structure as discussed in Sect. 2.4.3 follows immediately.

The GFSR and Twisted GFSR

2.4.5

Here we take A as the $pq \times pq$ matrix

$$A = \begin{pmatrix} & & & I_p & S \\ & & & I_p & \\ & & & I_p & \\ & & & \ddots & \\ & & & & I_p \end{pmatrix}$$

for some positive integers p and q , where I_p is the $p \times p$ identity matrix, S is a $p \times p$ matrix, and the matrix I_p on the first line is in columns $(r-1)p+1$ to rp for some positive integer r . Often, $w = p$ and B contains the first w lines

of the $pq \times pq$ identity matrix. If S is also the identity matrix, the generator thus obtained is the trinomial-based *generalized feedback shift register* (GFSR), for which x_i is obtained by a bitwise exclusive-or of x_{i-r} and x_{i-q} . This gives a very fast RNG, but its period length cannot exceed $2^q - 1$, because each bit of x_i follows the same binary recurrence of order $k = q$, with characteristic polynomial $P(z) = z^q - z^{q-r} - 1$.

More generally, we can define x_i as the bitwise exclusive-or of $x_{i-r_1}, x_{i-r_2}, \dots, x_{i-r_d}$ where $r_d = q$, so that each bit of x_i follows a recurrence in \mathbb{F}_2 whose characteristic polynomial $P(z)$ has $d + 1$ nonzero terms. However, the period length is still bounded by $2^q - 1$, whereas considering the pq -bit state, we should rather expect a period length close to 2^{pq} . This was the main motivation for the *twisted GFSR* (TGFSR) generator. In the original version introduced by Matsumoto and Kurita (1992), $w = p$ and the matrix S is defined as the transpose of A_0 in (2.16), with k replaced by p . The characteristic polynomial of A is then $P(z) = P_S(z^q + z^m)$, where $P_S(z) = z^p - a_p z^{p-1} - \dots - a_1$ is the characteristic polynomial of S , and its degree is $k = pq$. If the parameters are selected so that $P(z)$ is primitive over \mathbb{F}_2 , then the TGFSR has period length $2^k - 1$. Matsumoto and Kurita (1994) pointed out important weaknesses of the original TGFSR and proposed an improved version that uses a well-chosen matrix B whose lines differ from those of the identity. The operations implemented by this matrix are called *tempering* and their purpose is to improve the uniformity of the points produced by the RNG. The *Mersenne twister* (Matsumoto and Nishimura, 1998; Nishimura, 2000) is a variant of the TGFSR where k is slightly less than pq and can be a prime number. A specific instance proposed by Matsumoto and Nishimura (1998) is fast, robust, has the huge period length of $2^{19937} - 1$, and has become quite popular.

In the *multiple recursive matrix method* of Niederreiter (1995), the first row of $p \times p$ matrices in A contains arbitrary matrices. However, a fast implementation is possible only when these matrices are sparse and have a special structure.

2.4.6 Combined Linear Generators Over \mathbb{F}_2

Many of the best generators based on linear recurrences over \mathbb{F}_2 are constructed by combining the outputs of two or more RNGs having a simple structure. The idea is the same as for MRGs: select simple components that can run fast but such that their combination has a more complicated structure and highly-uniform sets Ψ_I for the sets I considered important.

Consider J distinct recurrences of the form (2.12)–(2.13), where the j th recurrence has parameters $(k, w, A, B) = (k_j, w, A_j, B_j)$ and state $x_{j,i}$ at step i , for $j = 1, \dots, J$. The output of the combined generator at step i is defined by

$$y_i = B_1 x_{1,i} \oplus \dots \oplus B_J x_{J,i},$$

$$u_i = \sum_{\ell=1}^w y_{i,\ell-1} 2^{-\ell},$$

where \oplus denotes the bitwise exclusive-or operation. One can show (Tezuka, 1995) that the period length ρ of this combined generator is the least common multiple of the period lengths ρ_j of its components. Moreover, this combined generator is equivalent to the generator (2.12)–(2.14) with $k = k_1 + \dots + k_j$, $A = \text{diag}(A_1, \dots, A_j)$, and $B = (B_1, \dots, B_j)$.

With this method, by selecting the parameters carefully, the combination of LFSR generators with characteristic polynomials $P_1(z), \dots, P_j(z)$ gives yet another LFSR with characteristic polynomial $P(z) = P_1(z) \cdots P_j(z)$ and period length equal to the product of the period lengths of the components (Tezuka and L'Ecuyer, 1991; Wang and Compagner, 1993; L'Ecuyer, 1996b; Tezuka, 1995). Tables and fast implementations of maximally equidistributed combined LFSR generators are given in L'Ecuyer (1996b).

The TGFSR and Mersenne twister generators proposed in Matsumoto and Kurita (1994), Matsumoto and Nishimura (1998) and Nishimura (2000) cannot be maximally equidistributed. However, concrete examples of maximally equidistributed combined TGFSR generators with period lengths near 2^{466} and 2^{1250} are given in L'Ecuyer and Panneton (2002). These generators have the additional property that the resolution gaps δ_I are zero for a class of small sets I with indices not too far apart.

Nonlinear RNGs

All RNGs discussed so far are based on linear recurrences and their structure may be deemed too regular. There are at least two ways of getting rid of this regular linear structure: (1) use a nonlinear transition function f or (2) keep the transition function linear but use a nonlinear output function g . Several types of nonlinear RNGs have been proposed over the years; see, e.g., Blum et al. (1986), Eichenauer-Herrmann (1995), Eichenauer-Herrmann et al. (1997), Hellekalek and Wegenkittl (2003), Knuth (1998), L'Ecuyer (1994), Niederreiter and Shparlinski (2002), and Tezuka (1995). Their nonlinear mappings are defined in various ways by multiplicative inversion in a finite field, quadratic and cubic functions in the finite ring of integers modulo m , and other more complicated transformations. Many of them have output sequences that tend to behave much like i.i.d. $U(0, 1)$ sequences even over their entire period length, in contrast with “good” linear RNGs, whose point sets Ψ_t are much more regular than typical random points (Eichenauer-Herrmann et al., 1997; L'Ecuyer and Hellekalek, 1998; L'Ecuyer and Granger-Piché, 2003; Niederreiter and Shparlinski, 2002). On the other hand, their statistical properties have been analyzed only empirically or via asymptotic theoretical results. For specific nonlinear RNGs, the uniformity of the point sets Ψ_t is very difficult to measure theoretically. Moreover, the nonlinear RNGs are generally significantly slower than the linear ones. The RNGs recommended for cryptology are all nonlinear.

An interesting idea for adding nonlinearity without incurring an excessive speed penalty is to combine a small nonlinear generator with a fast long-period linear

one (Aiello et al., 1998; L'Ecuyer and Granger-Piché, 2003). L'Ecuyer and Granger-Piché (2003) show how to do this while ensuring theoretically the good uniformity properties of Ψ_t for the combined generator. A very fast implementation can be achieved by using precomputed tables for the nonlinear component. Empirical studies suggest that mixed linear-nonlinear combined generators are more robust than the linear ones with respect to statistical tests, because of their less regular structure.

Several authors have proposed various ways of combining RNGs to produce streams of random numbers with less regularity and better “randomness” properties; see, e.g., Collings (1987), Knuth (1998), Gentle (2003), Law and Kelton (2000), L'Ecuyer (1994), Fishman (1996), Marsaglia (1985), and other references given there. This includes *shuffling* the output sequence of one generator using another one (or the same one), alternating between several streams, or just adding them in different ways. Most of these techniques are heuristics. They usually improve the uniformity (empirically), but they can also make it worse. For *random variables* in the mathematical sense, certain types of combinations (e.g., addition modulo 1) can *provably* improve the uniformity, and some authors have used this fact to argue that combined RNGs are provably better than their components alone (Brown and Solomon, 1979; Deng and George, 1990; Marsaglia, 1985; Gentle, 2003), but this argument is faulty because the output sequences of RNGs are deterministic, not sequences of independent random variables. To assess the quality of a combined generator, one must analyze the mathematical structure of the combined generator itself rather than the structure of its components (L'Ecuyer, 1996b, 1996a, 1998; L'Ecuyer and Granger-Piché, 2003; Tezuka, 1995).

2.6

Examples of Statistical Tests

As mentioned earlier, a statistical test for RNGs is defined by a random variable X whose distribution under \mathcal{H}_0 can be well approximated. When X takes the value x , we define the right and left *p-values* of the test by

$$p_R = P[X \geq x \mid \mathcal{H}_0] \quad \text{and} \quad p_L = P[X \leq x \mid \mathcal{H}_0] .$$

When testing RNGs, there is no need to prespecify the level of the test. If any of the right or left *p-value* is extremely close to zero, e.g., less than 10^{-15} , then it is clear that \mathcal{H}_0 (and the RNG) must be rejected. When a *suspicious p-value* is obtained, e.g., near 10^{-2} or 10^{-3} , one can just repeat this particular test a few more times, perhaps with a larger sample size. Almost always, things will then clarify.

Most tests are defined by partitioning the possible realizations of $(u_0, \dots, u_{\tau-1})$ into a finite number of subsets (where the integer τ can be random or deterministic), computing the probability p_j of each subset j under \mathcal{H}_0 , and measuring the discrepancy between these probabilities and empirical frequencies from realizations simulated by the RNG.

A special case that immediately comes to mind is to take $\tau = t$ (a constant) and cut the interval $[0, 1)$ into d equal segments for some positive integer d , in order to partition the hypercube $[0, 1)^t$ into $k = d^t$ subcubes of volume $1/k$. We then generate n points $\mathbf{u}_i = (u_{i1}, \dots, u_{i+t-1}) \in [0, 1)^t$, for $i = 0, \dots, n-1$, and count the number N_j of points falling in subcube j , for $j = 0, \dots, k-1$. Any measure of distance (or divergence) between the numbers N_j and their expectations n/k can define a test statistic X . The tests thus defined are generally called *serial tests* of uniformity (Knuth, 1998; L'Ecuyer et al., 2002b). They can be *sparse* (if $n/k \ll 1$), or *dense* (if $n/k \gg 1$), or somewhere in between. There are also *overlapping* versions, where the points are defined by $\mathbf{u}_i = (u_i, \dots, u_{i+t-1})$ for $i = 0, \dots, n-1$ (they have overlapping coordinates).

Special instances for which the distribution of X under \mathcal{H}_0 is well-known are the chi-square, the (negative) empirical entropy, and the number of collisions (L'Ecuyer and Hellekalek, 1998; L'Ecuyer et al., 2002b; Read and Cressie, 1988). For the latter, the test statistic X is the number of times a point falls in a subcube that already had a point in it. Its distribution under \mathcal{H}_0 is approximately Poisson with mean $\lambda_1 = n^2/(2k)$, if n is large and λ_1 is not too large.

A variant is the *birthday spacings* test, defined as follows (Marsaglia, 1985; Knuth, 1998; L'Ecuyer and Simard, 2001). Let $I_{(1)} \leq \dots \leq I_{(n)}$ be the numbers of the subcubes that contain the points, sorted by increasing order. Define the *spacings* $S_j = I_{(j+1)} - I_{(j)}$, for $j = 1, \dots, n-1$, and let X be the number of collisions between these spacings. Under \mathcal{H}_0 , X is approximately Poisson with mean $\lambda_2 = n^3/(4k)$, if n is large and λ_2 not too large.

Consider now a MRG, for which Ψ_t has a regular lattice structure. Because of this regularity the points of Ψ_t will tend to be more evenly distributed among the subcubes than random points. For a well-chosen k and large enough n , we expect the collision test to detect this: it is likely that there will be too few collisions. In fact, the same applies to any RNG whose set Ψ_t is very evenly distributed. When a birthday spacings test with a very large k is applied to a MRG, the numbers of the subcubes that contain one point of Ψ_t tend to be too evenly spaced and the test detects this by finding too many collisions.

These specific interactions between the test and the structure of the RNG lead to systematic patterns in the p -values of the tests. To illustrate this, suppose that we take k slightly larger than the cardinality of Ψ_t (so $k \approx \varrho$) and that due to the excessive regularity, no collision is observed in the collision test. The left p -value will then be $p_L \approx P[X \leq 0 \mid X \sim \text{Poisson}(\lambda_1)] = \exp[-n^2/(2k)]$. For this p -value to be smaller than a given ε , we need a sample size n proportional to the square root of the period length ϱ . And after that, p_L decreases exponentially fast in n^2 .

Extensive experiments with LCGs, MRGs, and LFSR generators confirms that this is actually what happens with these RNGs (L'Ecuyer and Hellekalek, 1998; L'Ecuyer, 2001; L'Ecuyer et al., 2002b). For example, if we take $\varepsilon = 10^{-15}$ and define n_0 as the minimal sample size n for which $p_L < \varepsilon$, we find that $n_0 \approx 16\varrho^{1/2}$ (plus some noise) for LCGs that behave well in the spectral test as well as for LFSR generators. For the birthday spacings test, the rule for LCGs is $n_0 \approx 16\varrho^{1/3}$ instead (L'Ecuyer and Simard, 2001). So to be safe with respect to these tests, the period

length φ must be so large that generating more than $\varphi^{1/3}$ numbers is practically unfeasible. This certainly disqualifies all LCGs with modulus smaller than 2^{100} or so.

Other types of tests for RNGs include tests based on the closest pairs of points among n points generated in the hypercube, tests based on random walks on the real line or over the integers, tests based on the linear complexity of a binary sequence, tests based on the simulation of dices or poker hands, and many others (Gentle, 2003; Knuth, 1998; L'Ecuyer and Simard, 2002; Marsaglia, 1996; Rukhin et al., 2001; Vattulainen et al., 1995).

When testing RNGs, there is no specific alternative hypothesis to \mathcal{H}_0 . Different tests are needed to detect different types of departures from \mathcal{H}_0 . *Test suites* for RNGs include a selection of tests, with predetermined parameters and sample sizes. The best known are probably DIEHARD (Marsaglia, 1996) and the NIST test suite (Rukhin et al., 2001). The library *TestU01* (L'Ecuyer and Simard, 2002) implements a large selection of tests in the C language and provides a variety of test suites, some designed for i.i.d. $U(0, 1)$ output sequences and others for strings of bits.

Available Software and Recommendations

2.7

When we apply test suites to RNGs currently found in commercial software (statistical and simulation software, spreadsheets, etc.), we find that many of them fail the tests spectacularly (L'Ecuyer, 1997,2001). There is no reason to use these poor RNGs, because there are also several good ones that are fast, portable, and pass all these test suites with flying colors. Among them we recommend, for example, the combined MRGs, combined LFSRs, and Mersenne twisters proposed in L'Ecuyer (1999c,1999a), L'Ecuyer and Panneton (2002), Matsumoto and Nishimura (1998), and Nishimura (2000).

A convenient object-oriented software package with multiple streams and sub-streams of random numbers, is described in L'Ecuyer et al. (2002a) and is available in Java, C, and C++, at <http://www.iro.umontreal.ca/~lecuyer>.

2.8

Non-uniform Random Variate Generation

Like for the uniform case, non-uniform variate generation often involves approximations and compromises. The first requirement is, of course, *correctness*. This does not mean that the generated random variate X must always have *exactly* the required distribution, because this would sometimes be much too costly or even impossible. But we must have a *good approximation* and, preferably, some understanding of the quality of that approximation. *Robustness* is also important:

when the accuracy depends on the parameters of the distribution, it must be good *uniformly* over the entire range of parameter values that we are interested in.

The method must also be *efficient* both in terms of speed and memory usage. Often, it is possible to increase the speed by using more memory (e.g. for larger precomputed tables) or by relaxing the accuracy requirements. Some methods need a one-time setup to compute constants and construct tables. The setup time can be significant but may be well worth spending if it is amortized by a large number of subsequent calls to the generator. For example, it makes sense to invest in a more extensive setup if we plan to make a million calls to a given generator than if we expect to make only a few calls, assuming that this investment can improve the speed of the generator sufficiently.

In general, compromises must be made between simplicity of the algorithm, quality of the approximation, robustness with respect to the distribution parameters, and efficiency (generation speed, memory requirements, and setup time).

In many situations, compatibility with variance reduction techniques is another important issue (Bratley et al., 1987; Law and Kelton, 2000). We may be willing to sacrifice the speed of the generator to preserve inversion, because the gain in efficiency obtained via the variance reduction methods may more than compensate (sometimes by orders of magnitude) for the slightly slower generator.

Inversion

2.8.1

The inversion method, defined in the introduction, should be the method of choice for generating non-uniform random variates in a majority of situations. The fact that $X = F^{-1}(U)$ is a monotone (non-decreasing) function of U makes this method compatible with important variance reduction techniques such as common random numbers, antithetic variates, latin hypercube sampling, and randomized quasi-Monte Carlo methods (Bratley et al., 1987; Law and Kelton, 2000; L'Ecuyer and Lemieux, 2000).

For some distributions, an analytic expression can be obtained for the inverse distribution function F^{-1} and inversion can be easily implemented. As an example, consider the *Weibull* distribution function with parameters $\alpha > 0$ and $\beta > 0$, defined by $F(x) = 1 - \exp[-(x/\beta)^\alpha]$ for $x > 0$. It is easy to see that $F^{-1}(U) = \beta[-\ln(1-U)]^{1/\alpha}$. For $\alpha = 1$, we have the special case of the exponential distribution with mean β .

For an example of a simple discrete distribution, suppose that $P[X = i] = p_i$ where $p_0 = 0.6$, $p_1 = 0.3$, $p_2 = 0.1$, and $p_i = 0$ elsewhere. The inversion method in this case will return 0 if $U < 0.6$, 1 if $0.6 \leq U < 0.9$, and 2 if $U \geq 0.9$. For the discrete uniform distribution over $\{0, \dots, k-1\}$, return $X = \lfloor kU \rfloor$. As another example, let X have the *geometric* distribution with parameter p , so $P[X = x] = p(1-p)^x$ for $x = 0, 1, 2, \dots$, where $0 < p < 1$. Then, $F(x) = 1 - (1-p)^{\lfloor x+1 \rfloor}$ for $x \geq 0$ and one can show that $X = F^{-1}(U) = \lceil \ln(1-U)/\ln(1-p) \rceil - 1$.

There are other distributions (e.g., the normal, Student, chi-square) for which there is no closed-form expression for F^{-1} but good numerical approximations are available (Bratley et al., 1987; Gentle, 2003; Marsaglia et al., 1994). When the

distribution has only scale and location parameters, we need to approximate F^{-1} only for a standardized version of the distribution. For the normal distribution, for example, it suffices to have an efficient method for evaluating the inverse distribution function of a $N(0, 1)$ random variable Z , since a normal with mean μ and variance σ^2 can be generated by $X = \sigma Z + \mu$. When shape parameters are involved (e.g., the gamma and beta distributions), things are more complicated because F^{-1} then depends on the parameters in a more fundamental manner.

Hörmann and Leydold (2003) propose a general adaptive and automatic method that constructs a highly accurate Hermite interpolation method of F^{-1} . In a one-time setup, their method produces tables for the interpolation points and coefficients. Random variate generation using these tables is then quite fast.

A less efficient but simpler way of implementing inversion when a method is available for computing F is via binary search (Cheng, 1998). If the density is also available and if it is unimodal with known mode, a Newton–Raphson iteration method can advantageously replace the binary search (Cheng, 1998; Devroye, 1986).

To implement inversion for general discrete distributions, sequential search and binary search with look-up tables are the standard methods (Bratley et al., 1987; Cheng, 1998). For a discrete distribution over the values $x_1 < \dots < x_k$, one first tabulates the pairs $(x_i, F(x_i))$, where $F(x_i) = P[X \leq x_i]$ for $i = 1, \dots, k$. To generate X , it then suffices to generate $U \sim U(0, 1)$, find $I = \min\{i \mid F(x_i) \geq U\}$, and return $X = x_I$. The following algorithms do that.

Sequential search (needs $O(k)$ iterations in the worst case);

```
generate  $U \sim U(0, 1)$ ; let  $i = 1$ ;
while  $F(x_i) < U$  do  $i = i + 1$ ;
return  $x_i$ .
```

Binary search (needs $O(\log k)$ iterations in the worst case);

```
generate  $U \sim U(0, 1)$ ; let  $L = 0$  and  $R = k$ ;
while  $L < R - 1$  do
   $m = \lfloor (L + R)/2 \rfloor$ ;
  if  $F(x_m) < U$  then  $L = m$  else  $R = m$ ;
/* Invariant: at this stage, the index  $I$  is in  $\{L + 1, \dots, R\}$ . */
return  $x_R$ .
```

These algorithms can be modified in many different ways. For example, if $k = \infty$, in the binary search, one can start with an arbitrary value of R , double it until $F(x_R) \geq U$, and start the algorithm with this R and $L = R/2$. Of course, only a finite portion of the table (a portion that contains most of the probability mass) would be precomputed in this case, the other values can be computed only when needed. This can also be done if k is finite but large.

Another class of techniques use indexing or buckets to speed up the search (Chen and Asau, 1974; Bratley et al., 1987; Devroye, 1986). For example, one can partition the interval $(0, 1)$ into c subintervals of equal sizes and use (pre-tabulated) initial values of (L, R) that depend on the subinterval in which U falls. For the subinterval $[j/c, (j+1)/c)$ the values of L and R would be $L_j = F^{-1}(j/c)$ and $R_j = F^{-1}((j+1)/c)$,

for $j = 0, \dots, c-1$. The subinterval number that corresponds to a given U is simply $J = \lfloor cU \rfloor$. Once we know that subinterval, we can search it by linear or binary search. With a larger value of c the search is faster (on the average) but the setup is more costly and a larger amount of memory is needed. So a compromise must be made depending on the situation (e.g., the value of k , the number of variates we expect to generate, etc.). For $c = 1$, we recover the basic sequential and binary search algorithms given above. A well-implemented indexed search with a large enough c is usually competitive with the alias method (described in the next paragraph). A combined indexed/binary search algorithm is given below. An easy adaptation gives the combined indexed/sequential search, which is generally preferable when k/c is small, because it has smaller overhead.

Indexed search (combined with binary search);

```
generate  $U \sim U(0, 1)$ ;  let  $J = \lfloor cU \rfloor, L = L_J$ , and  $R = R_J$ ;
while  $L < R - 1$  do
   $m = \lfloor (L + R)/2 \rfloor$ ;
  if  $F(x_m) < U$  then  $L = m$  else  $R = m$ ;
return  $x_R$ .
```

These search methods are also useful for piecewise-linear (or piecewise-polynomial) distribution functions. Essentially, it suffices to add an interpolation step at the end of the algorithm, after the appropriate linear (or polynomial) piece has been determined (Bratley et al., 1987).

Finally, the stochastic model itself can sometimes be selected in a way that makes inversion easier. For example, one can fit a parametric, highly-flexible, and easily computable inverse distribution function F^{-1} to the data, directly or indirectly (Nelson and Yamnitsky, 1998; Wagner and Wilson, 1996).

There are situations where *speed* is important and where non-inversion methods are appropriate. In forthcoming subsections, we outline the main non-inversion methods.

The Alias Method

 2.8.2

Sequential and binary search require $O(k)$ and $O(\log k)$ time, respectively, in the worst case, to generate a random variate X by inversion over the finite set $\{x_1, \dots, x_k\}$. The *alias method* (Walker, 1974, 1977) can generate such a X in $O(1)$ time per variate, after a table setup that takes $O(k)$ time and space. On the other hand, it does not implement inversion, i.e., the transformation from U to X is not monotone.

To explain the idea, consider a bar diagram of the distribution, where each index i has a bar of height $p_i = P[X = x_i]$. The idea is to “equalize” the bars so that they all have height $1/k$, by cutting-off bar pieces and transferring them to other bars. This is done in a way that in the new diagram, each bar i contains one piece of size q_i (say) from the original bar i and one piece of size $1/k - q_i$ from another bar whose index j , denoted $A(i)$, is called the *alias* value of i . The setup procedure initializes two tables, A and R , where $A(i)$ is the alias value of i and

$R(i) = (i - 1)/k + q_i$. See Devroye (1986) and Law and Kelton (2000) for the details. To generate X , we generate $U \sim U[0, 1]$, define $I = \lceil kU \rceil$, and return $X = x_I$ if $U < R(I)$ and $X = x_{A(I)}$ otherwise.

There is a version of the alias method for continuous distributions, called the *acceptance-complement* method (Kronmal and Peterson, 1984; Devroye, 1986; Gentle, 2003). The idea is to decompose the density f of the target distribution as the convex combination of two densities f_1 and f_2 , $f = wf_1 + (1 - w)f_2$ for some real number $w \in (0, 1)$, in a way that $wf_1 \leq g$ for some other density g and so that it is easy to generate from g and f_2 . The algorithm works as follows: Generate X from density g and $U \sim U(0, 1)$; if $Ug(X) \leq wf_1(X)$ return X , otherwise generate a new X from density f_2 and return it.

2.8.3 Kernel Density Estimation and Generation

Instead of selecting a parametric distribution that is hard to invert and estimating the parameters, one can estimate the density via a *kernel density estimation* method for which random variate generation is very easy (Devroye, 1986; Hörmann and Leydold, 2000). In the case of a gaussian kernel, for example, one can generate variates simply by selecting one observation at random from the data and adding random noise generated from a normal distribution with mean zero. However, this method is *not* equivalent to inversion. Because of the added noise, selecting a larger observation does not necessarily guarantee a larger value for the generated variate.

2.8.4 The Rejection Method

Now suppose we want to generate X from a complicated density f . In fact f may be known only up to a multiplicative constant $\kappa > 0$, i.e., we know only κf . If we know f , we may just take $\kappa = 1$. We select another density r such that $\kappa f(x) \leq t(x) \stackrel{\text{def}}{=} ar(x)$ for all x for some constant a , and such that generating variates Y from the density r is easy. The function t is called a *hat function* or *majorizing function*. By integrating this inequality with respect to x on both sides, we find that $\kappa \leq a$. The following *rejection* method generates X with density f (von Neumann, 1951; Devroye, 1986; Evans and Swartz, 2000):

Rejection method;

```
repeat
  generate  $Y$  from the density  $r$  and  $U \sim U(0, 1)$ , independently;
until  $Ut(Y) \leq \kappa f(Y)$ ;
return  $X = Y$ .
```

The number R of turns into the “repeat” loop is one plus a geometric random variable with parameter κ/a , so $E[R] = a/\kappa$. Thus, we want $a/\kappa \geq 1$ to be as small as possible, i.e., we want to minimize the area between κf and t . There is generally a compromise between bringing a/κ close to 1 and keeping r simple.

When κf is expensive to compute, we can also use *squeeze functions* q_1 and q_2 that are faster to evaluate and such that $q_1(x) \leq \kappa f(x) \leq q_2(x) \leq t(x)$ for all x . To verify the condition $Ut(Y) \leq \kappa f(Y)$, we first check if $Ut(Y) \leq q_1(Y)$, in which case we accept Y immediately, otherwise we check if $Ut(Y) \geq q_2(Y)$, in which case we reject Y immediately. The value of $\kappa f(Y)$ must be computed only when $Ut(Y)$ falls between the two squeezes. Sequences of imbedded squeezes can also be used, where the primary ones are the least expensive to compute, the secondary ones are a little more expensive but closer to κf , etc.

It is common practice to transform the density f by a smooth increasing function T (e.g., $T(x) = \log x$ or $T(x) = -x^{-1/2}$) selected so that it is easier to construct good hat and squeeze functions (often piecewise linear) for the transformed density $T(f(\cdot))$. By transforming back to the original scale, we get hat and squeeze functions for f . This is the *transformed density rejection* method, which has several variants and extensions (Devroye, 1986; Evans and Swartz, 2000; Hörmann et al., 2004).

The rejection method works for discrete distributions as well; it suffices to replace densities by probability mass functions.

Thinning for Point Processes with Time-varying Rates

2.8.5

Thinning is a cousin of acceptance-rejection, often used for generating events from a non-homogeneous Poisson process. Suppose the process has rate $\lambda(t)$ at time t , with $\lambda(t) \leq \bar{\lambda}$ for all t , where $\bar{\lambda}$ is a finite constant. One can generate Poisson *pseudo-arrivals* at constant rate $\bar{\lambda}$ by generating interarrival times that are i.i.d. exponentials of mean $1/\bar{\lambda}$. Then, a pseudo-arrival at time t is accepted (becomes an arrival) with probability $\lambda(t)/\bar{\lambda}$ (i.e., if $U \leq \lambda(t)/\bar{\lambda}$, where U is an independent $U[0, 1]$), and rejected with probability $1 - \lambda(t)/\bar{\lambda}$. Non-homogeneous Poisson processes can also be generated by inversion (Bratley et al., 1987). The idea is to apply a nonlinear transformation to the time scale to make the process homogeneous with rate 1 in the new time scale. Arrival times are generated in this new time scale (which is easy), and then transformed back to the original time scale. The method can be adapted to other types of point processes with time-varying rates.

The Ratio-of-Uniforms Method

2.8.6

If f is a density over the real-line, κ an arbitrary positive constant, and the pair (U, V) has the uniform distribution over the set

$$\mathcal{C} = \left\{ (u, v) \in \mathbb{R}^2 \quad \text{such that} \quad 0 \leq u \leq \sqrt{\kappa f(v/u)} \right\},$$

then V/U has density f (Kinderman and Monahan, 1977; Devroye, 1986; Gentle, 2003). This interesting property can be exploited to generate X with density f : generate (U, V) uniformly over \mathcal{C} and return $X = V/U$. This is the *ratio-of-uniforms* method. The key issue is how do we generate a point uniformly over \mathcal{C} .

In the cases where this can be done efficiently, we immediately have an efficient way of generating X .

The most frequent approach for generating (U, V) uniformly over \mathcal{C} is the rejection method: Define a region \mathcal{C}_2 that contains \mathcal{C} and in which it is easy to generate a point uniformly (for example, a rectangular box or a polygonal region). To generate X , repeat: generate (U, V) uniformly over \mathcal{C}_2 , until it belongs to \mathcal{C} . Then return $X = V/U$. If there is another region \mathcal{C}_1 contained in \mathcal{C} and for which it is very fast to check if a point (U, V) is in \mathcal{C}_1 , this \mathcal{C}_1 can also be used as a squeeze to accelerate the verification that the point belongs to \mathcal{C} . Several special cases and refinements are described in Devroye (1986), Gentle (2003), Leydold (2000), and other references given there.

2.8.7 Composition and Convolution

Suppose F is a convex combination of several distributions, i.e., $F(x) = \sum_j p_j F_j(x)$, or more generally $F(x) = \int F_y(x) dH(y)$. To generate from F , one can generate $J = j$ with probability p_j (or Y from H), then generate X from F_j (or F_Y). This method, called the *composition algorithm*, is useful for generating from *compound* distributions such as the hyperexponential or from compound Poisson processes. It is also frequently used to design specialized algorithms for generating from complicated densities. The idea is to partition the area under the complicated density into pieces, where piece j has surface p_j . To generate X , first select a piece (choose piece j with probability p_j), then draw a random point uniformly over that piece and project it to the horizontal axis. If the partition is defined so that it is fast and easy to generate from the large pieces, then X will be returned very quickly most of the time. The rejection method with a squeeze is often used to generate from some of the pieces.

A dual method to composition is the *convolution method*, which can be used when $X = Y_1 + Y_2 + \dots + Y_n$, where the Y_i 's are independent with specified distributions. With this method, one just generates the Y_i 's and sums up. This requires at least n uniforms. Examples of random variables that can be expressed as sums like this include the hypoexponential, Erlang, and binomial distributions.

2.8.8 Other Special Techniques

Besides the general methods, several specialized and sometimes very elegant techniques have been designed for commonly used distributions such as the Poisson distribution with small mean, the normal (e.g., the Box-Muller method), for generating points uniformly on a k -dimensional sphere, for generating random permutations, and so on. Details can be found, e.g., in Bratley et al. (1987), Cheng (1998), Devroye (1986), Fishman (1996), Gentle (2003).

Recently, there has been an effort in developing *automatic* or *black box* algorithms for generating variates from an arbitrary (known) density, and reliable software that implements these methods (Hörmann and Leydold, 2000; Hörmann et al., 2004; Leydold and Hörmann, 2002; Leydold et al., 2002).

Acknowledgements. This work has been supported by the Natural Sciences and Engineering Research Council of Canada (NSERC) Grant No. ODGP0110050, NATEQ-Québec grant No. 02ER3218, and a Canada Research Chair to the author. Wolfgang Hörmann, Josef Leydold, François Panneton, and Richard Simard made helpful comments and corrections on an earlier draft. The author has been asked to write chapters on Random Number Generation for several handbooks and encyclopedia recently. Inevitably, there is a large amount of duplication between these chapters.

References

- Aiello, W., Rajagopalan, S., and Venkatesan, R. (1998). Design of practical and provably good random number generators. *Journal of Algorithms*, 29(2): 358–389.
- Blum, L., Blum, M., and Schub, M. (1986). A simple unpredictable pseudo-random number generator. *SIAM Journal on Computing*, 15(2):364–383.
- Bratley, P., Fox, B. L., and Schrage, L. E. (1987). *A Guide to Simulation*. Springer-Verlag, New York, second edition.
- Brown, M. and Solomon, H. (1979). On combining pseudorandom number generators. *Annals of Statistics*, 1:691–695.
- Chen, H. C. and Asau, Y. (1974). On generating random variates from an empirical distribution. *AIEE Transactions*, 6:163–166.
- Cheng, R. C. H. (1998). Random variate generation. In Banks, J., editor, *Handbook of Simulation*, pages 139–172. Wiley. Chap. 5.
- Collings, B. J. (1987). Compound random number generators. *Journal of the American Statistical Association*, 82(398):525–527.
- Conway, J. H. and Sloane, N. J. A. (1999). *Sphere Packings, Lattices and Groups*. Grundlehren der Mathematischen Wissenschaften 290. Springer-Verlag, New York, 3rd edition.
- Couture, R. and L’Ecuyer, P. (1994). On the lattice structure of certain linear congruential sequences related to AWC/SWB generators. *Mathematics of Computation*, 62(206):798–808.
- Couture, R. and L’Ecuyer, P. (1996). Orbits and lattices for linear random number generators with composite moduli. *Mathematics of Computation*, 65(213):189–201.
- Couture, R. and L’Ecuyer, P. (1997). Distribution properties of multiply-with-carry random number generators. *Mathematics of Computation*, 66(218):591–607.
- Couture, R. and L’Ecuyer, P. (2000). Lattice computations for random numbers. *Mathematics of Computation*, 69(230):757–765.
- Deng, L.-Y. and George, E. O. (1990). Generation of uniform variates from several nearly uniformly distributed variables. *Communications in Statistics*, B19(1):145–154.
- Deng, L.-Y. and Lin, D. K. J. (2000). Random number generation for the new century. *The American Statistician*, 54(2):145–150.

- Deng, L.-Y. and Xu, H. (2003). A system of high-dimensional, efficient, long-cycle and portable uniform random number generators. *ACM Transactions on Modeling and Computer Simulation*, 13(4):299–309.
- Devroye, L. (1986). *Non-Uniform Random Variate Generation*. Springer-Verlag, New York.
- Eichenauer-Herrmann, J. (1995). Pseudorandom number generation by nonlinear methods. *International Statistical Reviews*, 63:247–255.
- Eichenauer-Herrmann, J., Herrmann, E., and Wegenkittl, S. (1997). A survey of quadratic and inversive congruential pseudorandom numbers. In Hellekalek, P., Larcher, G., Niederreiter, H., and Zinterhof, P., editors, *Monte Carlo and Quasi-Monte Carlo Methods in Scientific Computing*, volume 127 of *Lecture Notes in Statistics*, pages 66–97, New York. Springer.
- Evans, M. and Swartz, T. (2000). *Approximating Integrals via Monte Carlo and Deterministic Methods*. Oxford University Press, Oxford, UK.
- Fincke, U. and Pohst, M. (1985). Improved methods for calculating vectors of short length in a lattice, including a complexity analysis. *Mathematics of Computation*, 44:463–471.
- Fishman, G. S. (1996). *Monte Carlo: Concepts, Algorithms, and Applications*. Springer Series in Operations Research. Springer-Verlag, New York.
- Fushimi, M. (1983). Increasing the orders of equidistribution of the leading bits of the Tausworthe sequence. *Information Processing Letters*, 16:189–192.
- Gentle, J. E. (2003). *Random Number Generation and Monte Carlo Methods*. Springer, New York, second edition.
- Goresky, M. and Klapper, A. (2003). Efficient multiply-with-carry random number generators with maximal period. *ACM Transactions on Modeling and Computer Simulation*, 13(4):310–321.
- Hellekalek, P. and Larcher, G., editors (1998). *Random and Quasi-Random Point Sets*, volume 138 of *Lecture Notes in Statistics*. Springer, New York.
- Hellekalek, P. and Wegenkittl, S. (2003). Empirical evidence concerning AES. *ACM Transactions on Modeling and Computer Simulation*, 13(4):322–333.
- Hörmann, W. and Leydold, J. (2000). Automatic random variate generation for simulation input. In Joines, J. A., Barton, R. R., Kang, K., and Fishwick, P. A., editors, *Proceedings of the 2000 Winter Simulation Conference*, pages 675–682, Piscataway, NJ. IEEE Press.
- Hörmann, W. and Leydold, J. (2003). Continuous random variate generation by fast numerical inversion. *ACM Transactions on Modeling and Computer Simulation*, 13(4):347–362.
- Hörmann, W., Leydold, J., and Derflinger, G. (2004). *Automatic Nonuniform Random Variate Generation*. Springer-Verlag, Berlin.
- Kinderman, A. J. and Monahan, J. F. (1977). Computer generation of random variables using the ratio of uniform deviates. *ACM Transactions on Mathematical Software*, 3:257–260.
- Knuth, D. E. (1998). *The Art of Computer Programming, Volume 2: Seminumerical Algorithms*. Addison-Wesley, Reading, Mass., third edition.

- Kronmal, R. A. and Peterson, A. V. (1984). An acceptance-complement analogue of the mixture-plus-acceptance-rejection method for generating random variables. *ACM Transactions on Mathematical Software*, 10:271–281.
- Lagarias, J. C. (1993). Pseudorandom numbers. *Statistical Science*, 8(1):31–39.
- Law, A. M. and Kelton, W. D. (2000). *Simulation Modeling and Analysis*. McGraw-Hill, New York, third edition.
- L'Ecuyer, P. (1990). Random numbers for simulation. *Communications of the ACM*, 33(10):85–97.
- L'Ecuyer, P. (1994). Uniform random number generation. *Annals of Operations Research*, 53:77–120.
- L'Ecuyer, P. (1996a). Combined multiple recursive random number generators. *Operations Research*, 44(5):816–822.
- L'Ecuyer, P. (1996b). Maximally equidistributed combined Tausworthe generators. *Mathematics of Computation*, 65(213):203–213.
- L'Ecuyer, P. (1997). Bad lattice structures for vectors of non-successive values produced by some linear recurrences. *INFORMS Journal on Computing*, 9(1):57–60.
- L'Ecuyer, P. (1998). Random number generation. In Banks, J., editor, *Handbook of Simulation*, pages 93–137. Wiley. Chap. 4.
- L'Ecuyer, P. (1999a). Good parameters and implementations for combined multiple recursive random number generators. *Operations Research*, 47(1):159–164.
- L'Ecuyer, P. (1999b). Tables of linear congruential generators of different sizes and good lattice structure. *Mathematics of Computation*, 68(225):249–260.
- L'Ecuyer, P. (1999c). Tables of maximally equidistributed combined LFSR generators. *Mathematics of Computation*, 68(225):261–269.
- L'Ecuyer, P. (2001). Software for uniform random number generation: Distinguishing the good and the bad. In *Proceedings of the 2001 Winter Simulation Conference*, pages 95–105, Piscataway, NJ. IEEE Press.
- L'Ecuyer, P. (2004). Polynomial integration lattices. In Niederreiter, H., editor, *Monte Carlo and Quasi-Monte Carlo Methods 2002*, pages 73–98, Berlin. Springer-Verlag.
- L'Ecuyer, P. and Andres, T. H. (1997). A random number generator based on the combination of four LCGs. *Mathematics and Computers in Simulation*, 44:99–107.
- L'Ecuyer, P., Blouin, F., and Couture, R. (1993). A search for good multiple recursive random number generators. *ACM Transactions on Modeling and Computer Simulation*, 3(2):87–98.
- L'Ecuyer, P. and Côté, S. (1991). Implementing a random number package with splitting facilities. *ACM Transactions on Mathematical Software*, 17(1):98–111.
- L'Ecuyer, P. and Couture, R. (1997). An implementation of the lattice and spectral tests for multiple recursive linear random number generators. *INFORMS Journal on Computing*, 9(2):206–217.
- L'Ecuyer, P. and Granger-Piché, J. (2003). Combined generators with components from different families. *Mathematics and Computers in Simulation*, 62:395–404.

- L'Ecuyer, P. and Hellekalek, P. (1998). Random number generators: Selection criteria and testing. In Hellekalek, P. and Larcher, G., editors, *Random and Quasi-Random Point Sets*, volume 138 of *Lecture Notes in Statistics*, pages 223–265. Springer, New York.
- L'Ecuyer, P. and Lemieux, C. (2000). Variance reduction via lattice rules. *Management Science*, 46(9):1214–1235.
- L'Ecuyer, P. and Lemieux, C. (2002). Recent advances in randomized quasi-Monte Carlo methods. In Dror, M., L'Ecuyer, P., and Szidarovszki, F., editors, *Modeling Uncertainty: An Examination of Stochastic Theory, Methods, and Applications*, pages 419–474. Kluwer Academic Publishers, Boston.
- L'Ecuyer, P. and Panneton, F. (2002). Construction of equidistributed generators based on linear recurrences modulo 2. In Fang, K.-T., Hickernell, F. J., and Niederreiter, H., editors, *Monte Carlo and Quasi-Monte Carlo Methods 2000*, pages 318–330. Springer-Verlag, Berlin.
- L'Ecuyer, P. and Proulx, R. (1989). About polynomial-time “unpredictable” generators. In *Proceedings of the 1989 Winter Simulation Conference*, pages 467–476. IEEE Press.
- L'Ecuyer, P. and Simard, R. (1999). Beware of linear congruential generators with multipliers of the form $a = \pm 2^q \pm 2^r$. *ACM Transactions on Mathematical Software*, 25(3):367–374.
- L'Ecuyer, P. and Simard, R. (2001). On the performance of birthday spacings tests for certain families of random number generators. *Mathematics and Computers in Simulation*, 55(1–3):131–137.
- L'Ecuyer, P. and Simard, R. (2002). *TestU01: A Software Library in ANSI C for Empirical Testing of Random Number Generators*. Software user's guide.
- L'Ecuyer, P., Simard, R., Chen, E. J., and Kelton, W. D. (2002a). An object-oriented random-number package with many long streams and substreams. *Operations Research*, 50(6):1073–1075.
- L'Ecuyer, P., Simard, R., and Wegenkittl, S. (2002b). Sparse serial tests of uniformity for random number generators. *SIAM Journal on Scientific Computing*, 24(2):652–668.
- L'Ecuyer, P. and Tezuka, S. (1991). Structural properties for two classes of combined random number generators. *Mathematics of Computation*, 57(196):735–746.
- L'Ecuyer, P. and Touzin, R. (2000). Fast combined multiple recursive generators with multipliers of the form $a = \pm 2^q \pm 2^r$. In Joines, J. A., Barton, R. R., Kang, K., and Fishwick, P. A., editors, *Proceedings of the 2000 Winter Simulation Conference*, pages 683–689, Piscataway, NJ. IEEE Press.
- L'Ecuyer, P. and Touzin, R. (2004). On the Deng-Lin random number generators and related methods. *Statistics and Computing*, 14:5–9.
- Leeb, H. (1995). Random numbers for computer simulation. Master's thesis, University of Salzburg.
- Lemieux, C. and L'Ecuyer, P. (2003). Randomized polynomial lattice rules for multivariate integration and simulation. *SIAM Journal on Scientific Computing*, 24(5):1768–1789.

- Leydold, J. (2000). Automatic sampling with the ratio-of-uniform method. *ACM Transactions on Mathematical Software*, 26(1):78–98.
- Leydold, J. and Hörmann, W. (2002). *UNURAN—A Library for Universal Non-Uniform Random Number Generators*. Available at <http://statistik.wu-wien.ac.at/unuran>.
- Leydold, J., Janka, E., and Hörmann, W. (2002). Variants of transformed density rejection and correlation induction. In Fang, K.-T., Hickernell, F. J., and Niederreiter, H., editors, *Monte Carlo and Quasi-Monte Carlo Methods 2000*, pages 345–356, Berlin. Springer-Verlag.
- Luby, M. (1996). *Pseudorandomness and Cryptographic Applications*. Princeton University Press, Princeton.
- Lüscher, M. (1994). A portable high-quality random number generator for lattice field theory simulations. *Computer Physics Communications*, 79:100–110.
- Marsaglia, G. (1985). A current view of random number generators. In *Computer Science and Statistics, Sixteenth Symposium on the Interface*, pages 3–10, North-Holland, Amsterdam. Elsevier Science Publishers.
- Marsaglia, G. (1996). The Marsaglia random number CDROM including the DIEHARD battery of tests of randomness. See <http://stat.fsu.edu/pub/diehard>.
- Marsaglia, G. and Zaman, A. (1991). A new class of random number generators. *The Annals of Applied Probability*, 1:462–480.
- Marsaglia, G., Zaman, A., and Marsaglia, J. C. W. (1994). Rapid evaluation of the inverse normal distribution function. *Statistic and Probability Letters*, 19:259–266.
- Matsumoto, M. and Kurita, Y. (1992). Twisted GFSR generators. *ACM Transactions on Modeling and Computer Simulation*, 2(3):179–194.
- Matsumoto, M. and Kurita, Y. (1994). Twisted GFSR generators II. *ACM Transactions on Modeling and Computer Simulation*, 4(3):254–266.
- Matsumoto, M. and Kurita, Y. (1996). Strong deviations from randomness in m -sequences based on trinomials. *ACM Transactions on Modeling and Computer Simulation*, 6(2):99–106.
- Matsumoto, M. and Nishimura, T. (1998). Mersenne twister: A 623-dimensionally equidistributed uniform pseudo-random number generator. *ACM Transactions on Modeling and Computer Simulation*, 8(1):3–30.
- Nelson, B. L. and Yamnitsky, M. (1998). Input modeling tools for complex problems. In *Proceedings of the 1998 Winter Simulation Conference*, pages 105–112, Piscataway, NJ. IEEE Press.
- Niederreiter, H. (1992). *Random Number Generation and Quasi-Monte Carlo Methods*, volume 63 of *SIAM CBMS-NSF Regional Conference Series in Applied Mathematics*. SIAM, Philadelphia.
- Niederreiter, H. (1995). The multiple-recursive matrix method for pseudorandom number generation. *Finite Fields and their Applications*, 1:3–30.
- Niederreiter, H. and Shparlinski, I. E. (2002). Recent advances in the theory of nonlinear pseudorandom number generators. In Fang, K.-T., Hickernell, F. J., and Niederreiter, H., editors, *Monte Carlo and Quasi-Monte Carlo Methods 2000*, pages 86–102, Berlin. Springer-Verlag.

- Nishimura, T. (2000). Tables of 64-bit Mersenne twisters. *ACM Transactions on Modeling and Computer Simulation*, 10(4):348–357.
- Panneton, F. and L'Ecuyer, P. (2004). Random number generators based on linear recurrences in F_2^w . In Niederreiter, H., editor, *Monte Carlo and Quasi-Monte Carlo Methods 2002*, pages 367–378, Berlin. Springer-Verlag.
- Read, T. R. C. and Cressie, N. A. C. (1988). *Goodness-of-Fit Statistics for Discrete Multivariate Data*. Springer Series in Statistics. Springer-Verlag, New York.
- Rukhin, A., Soto, J., Nechvatal, J., Smid, M., Barker, E., Leigh, S., Levenson, M., Vangel, M., Banks, D., Heckert, A., Dray, J., and Vo, S. (2001). A statistical test suite for random and pseudorandom number generators for cryptographic applications. NIST special publication 800-22, National Institute of Standards and Technology (NIST), Gaithersburg, Maryland, USA. See <http://csrc.nist.gov/rng/>.
- Tausworthe, R. C. (1965). Random numbers generated by linear recurrence modulo two. *Mathematics of Computation*, 19:201–209.
- Tezuka, S. (1995). *Uniform Random Numbers: Theory and Practice*. Kluwer Academic Publishers, Norwell, Mass.
- Tezuka, S. and L'Ecuyer, P. (1991). Efficient and portable combined Tausworthe random number generators. *ACM Transactions on Modeling and Computer Simulation*, 1(2):99–112.
- Tezuka, S., L'Ecuyer, P., and Couture, R. (1994). On the add-with-carry and subtract-with-borrow random number generators. *ACM Transactions of Modeling and Computer Simulation*, 3(4):315–331.
- Tootill, J. P. R., Robinson, W. D., and Eagle, D. J. (1973). An asymptotically random Tausworthe sequence. *Journal of the ACM*, 20:469–481.
- Vattulainen, I., Ala-Nissila, T., and Kankaala, K. (1995). Physical models as tests of randomness. *Physical Review E*, 52(3):3205–3213.
- von Neumann, J. (1951). Various techniques used in connection with random digits. In Householder, A. S. et al., editors, *The Monte Carlo Method*, number 12 in National Bureau of Standards Applied Mathematics Series, pages 36–38.
- Wagner, M. A. F. and Wilson, J. R. (1996). Using univariate Bézier distributions to model simulation input processes. *IIE Transactions*, 28:699–711.
- Walker, A. J. (1974). New fast method for generating discrete random numbers with arbitrary frequency distributions. *Electronic Letters*, 10:127–128.
- Walker, A. J. (1977). An efficient method for generating discrete random variables with general distributions. *ACM Transactions on Mathematical Software*, 3:253–256.
- Wang, D. and Compagner, A. (1993). On the use of reducible polynomials as random number generators. *Mathematics of Computation*, 60:363–374.
- Wegenkittl, S. and Matsumoto, M. (1999). Getting rid of correlations among pseudorandom numbers: Discarding versus tempering. *ACM Transactions on Modeling and Computer Simulation*, 9(3):282–294.
- Wu, P.-C. (1997). Multiplicative, congruential random number generators with multiplier $\pm 2^{k_1} \pm 2^{k_2}$ and modulus $2^p - 1$. *ACM Transactions on Mathematical Software*, 23(2):255–265.

Markov Chain Monte Carlo Technology

II.3

Siddhartha Chib

3.1	<i>Introduction</i>	72
	Organization	74
3.2	<i>Markov Chains</i>	74
	Definitions and Results	75
	Computation of Numerical Accuracy and Inefficiency Factor	78
3.3	<i>Metropolis–Hastings Algorithm</i>	79
	Convergence Results	82
	Example	83
	Multiple-Block M–H Algorithm	86
3.4	<i>The Gibbs Sampling Algorithm</i>	89
	The Algorithm	89
	Invariance of the Gibbs Markov Chain	91
	Sufficient Conditions for Convergence	91
	Example: Simulating a Truncated Multivariate Normal	92
3.5	<i>MCMC Sampling with Latent Variables</i>	92
3.6	<i>Estimation of Density Ordinates</i>	95
3.7	<i>Sampler Performance and Diagnostics</i>	96
3.8	<i>Strategies for Improving Mixing</i>	97
	Choice of Blocking	97
	Tuning the Proposal Density	98
	Other Strategies	98
3.9	<i>Concluding Remarks</i>	98

Introduction

In the past fifteen years computational statistics has been enriched by a powerful, somewhat abstract method of generating variates from a target probability distribution that is based on Markov chains whose stationary distribution is the probability distribution of interest. This class of methods, popularly referred to as Markov chain Monte Carlo methods, or simply MCMC methods, have been influential in the modern practice of Bayesian statistics where these methods are used to summarize the posterior distributions that arise in the context of the Bayesian prior-posterior analysis (Tanner and Wong, 1987; Gelfand and Smith, 1990; Smith and Roberts, 1993; Tierney, 1994; Besag et al., 1995; Chib and Greenberg, 1995, 1996; Gilks et al., 1996; Tanner, 1996; Gammerman, 1997; Robert and Casella, 1999; Carlin and Louis, 2000; Chen et al., 2000; Chib, 2001; Congdon, 2001; Liu, 2001; Robert, 2001; Gelman et al., 2003). MCMC methods have proved useful in practically all aspects of Bayesian inference, for example, in the context of prediction problems and in the computation of quantities, such as the marginal likelihood, that are used for comparing competing Bayesian models.

A central reason for the widespread interest in MCMC methods is that these methods are extremely general and versatile and can be used to sample univariate and multivariate distributions when other methods (for example classical methods that produce independent and identically distributed draws) either fail or are difficult to implement. The fact that MCMC methods produce dependent draws causes no substantive complications in summarizing the target distribution. For example, if $\{\boldsymbol{\psi}^{(1)}, \dots, \boldsymbol{\psi}^{(M)}\}$ are draws from a (say continuous) target distribution $\pi(\boldsymbol{\psi})$, where $\boldsymbol{\psi} \in \mathfrak{R}^d$, then the expectation of $h(\boldsymbol{\psi})$ under π can be estimated by the average

$$M^{-1} \sum_{j=1}^M h(\boldsymbol{\psi}^{(j)}) , \quad (3.1)$$

as in the case of random samples, because suitable laws of large numbers for Markov chains can be used to show that

$$M^{-1} \sum_{j=1}^M h(\boldsymbol{\psi}^{(j)}) \rightarrow \int_{\mathfrak{R}^d} h(\boldsymbol{\psi}) \pi(\boldsymbol{\psi}) d\boldsymbol{\psi} ,$$

as the simulation sample size M becomes large.

Another reason for the interest in MCMC methods is that, somewhat surprisingly, it is rather straightforward to construct one or more Markov chains whose limiting invariant distribution is the desired target distribution. One way to construct the appropriate Markov chain is by a method called the Metropolis method which was introduced by Metropolis et al. (1953) in connection with work related to the hydrogen bomb project. In this method, the Markov chain simulation is constructed by a recursive two step process. Given the current iterate $\boldsymbol{\psi}^{(j)}$, a pro-

positional value $\boldsymbol{\psi}'$ is drawn from a distribution $q(\boldsymbol{\psi}^{(j)}, \cdot)$, such that $\boldsymbol{\psi}'$ is symmetrically distributed about the current value $\boldsymbol{\psi}^{(j)}$. In the second step, this proposal value is accepted as the next iterate $\boldsymbol{\psi}^{(j+1)}$ of the Markov chain with probability

$$\alpha(\boldsymbol{\psi}^{(j)}, \boldsymbol{\psi}') = \min \left\{ 1, \frac{\pi(\boldsymbol{\psi}')}{\pi(\boldsymbol{\psi}^{(j)})} \right\}.$$

If the proposal value is rejected, then $\boldsymbol{\psi}^{(j+1)}$ is taken to be the current value. The method is simple to implement, even in multivariate settings, and was widely used by physicists in computational statistical mechanics and quantum field theory to sample the coordinates of a point in phase space. In those settings, and in subsequent statistical problems, it is helpful that the method can be implemented without knowledge of the normalizing constant of π since that constant cancels in the determination of the probability $\alpha(\boldsymbol{\psi}^{(j)}, \boldsymbol{\psi}')$.

The requirement that the proposal distribution be symmetric about the current value was relaxed by Hastings (1970). The resulting method, commonly called the Metropolis–Hastings (M–H) method, relies on the same two steps of the Metropolis method except that the probability of move is given by

$$\alpha(\boldsymbol{\psi}^{(j)}, \boldsymbol{\psi}') = \min \left\{ 1, \frac{\pi(\boldsymbol{\psi}') q(\boldsymbol{\psi}', \boldsymbol{\psi}^{(j)})}{\pi(\boldsymbol{\psi}^{(j)}) q(\boldsymbol{\psi}^{(j)}, \boldsymbol{\psi}')} \right\}$$

which clearly reduces to the Metropolis probability of move when the proposal distribution is symmetric in its arguments. Starting with an arbitrary value $\boldsymbol{\psi}^{(0)}$ in the support of the target distributions, iterations of this two step process produce the (correlated) sequence of values

$$\{\boldsymbol{\psi}^{(0)}, \boldsymbol{\psi}^{(1)}, \boldsymbol{\psi}^{(2)}, \dots\}.$$

Typically, a certain number of values (say n_0) at the start of this sequence are discarded and the subsequent (say) M values are used as variates from the target distribution.

In applications when the dimension of $\boldsymbol{\psi}$ is large it may be preferable to construct the Markov chain simulation by first grouping the variables $\boldsymbol{\psi}$ into smaller blocks. For simplicity suppose that two blocks are adequate and that $\boldsymbol{\psi}$ is written as $(\boldsymbol{\psi}_1, \boldsymbol{\psi}_2)$, with $\boldsymbol{\psi}_k \in \Omega_k \subseteq \mathfrak{R}^{d_k}$. In that case, the M–H chain can be constructed by

- updating $\boldsymbol{\psi}_1$ given $(\boldsymbol{\psi}_1^{(j)}, \boldsymbol{\psi}_2^{(j)})$ to produce $\boldsymbol{\psi}_1^{(j)}$ and then
- updating $\boldsymbol{\psi}_2$ given $(\boldsymbol{\psi}_1^{(j+1)}, \boldsymbol{\psi}_2^{(j)})$ to produce $\boldsymbol{\psi}_2^{(j+1)}$,

which completes one cycle through two sub-moves. Chib and Greenberg (1995) who emphasized and highlighted such M–H chains have referred to them as multiple-block M–H algorithms.

Despite the long vintage of the M–H method, the contemporary interest in MCMC methods was sparked by work on a related MCMC method, the Gibbs

sampling algorithm. The Gibbs sampling algorithm is one of the simplest Markov chain Monte Carlo algorithms and has its origins in the work of Besag (1974) on spatial lattice systems, Geman and Geman (1984) on the problem of image processing, and Tanner and Wong (1987) on missing data problems. The paper by Gelfand and Smith (1990) helped to demonstrate the value of the Gibbs algorithm for a range of problems in Bayesian analysis. In the Gibbs sampling method, the Markov chain is constructed by simulating the conditional distributions that are implied by $\pi(\boldsymbol{\psi})$. In particular, if $\boldsymbol{\psi}$ is split into two components $\boldsymbol{\psi}_1$ and $\boldsymbol{\psi}_2$, then the Gibbs method proceeds through the recursive sampling of the conditional distributions $\pi(\boldsymbol{\psi}_1|\boldsymbol{\psi}_2)$ and $\pi(\boldsymbol{\psi}_2|\boldsymbol{\psi}_1)$, where the most recent value of $\boldsymbol{\psi}_2$ is used in the first simulation and the most recent value of $\boldsymbol{\psi}_1$ in the second simulation. This method is most simple to implement when each conditional distribution is a known distribution that is easy to sample. As we show below, the Gibbs sampling method is a special case of the multiple block M–H algorithm.

3.1.1 Organization

The rest of the chapter is organized as follows. In Sect. 3.2 we summarize the relevant Markov chain theory that justifies simulation by MCMC methods. In particular, we provide the conditions under which discrete-time and continuous state space Markov chains satisfy a law of large numbers and a central limit theorem. The M–H algorithm is discussed in Sect. 3.3 followed by the Gibbs sampling algorithm in Sect. 3.4. Section 3.5 deals with MCMC methods with latent variables and Sect. 3.6 with ways of estimating the marginal densities based on the MCMC output. Issues related to sampler performance are considered in Sect. 3.7 and strategies for improving the mixing of the Markov chains in Sect. 3.8. Section 3.9 concludes with brief comments about new and emerging directions in MCMC methods.

3.2 Markov Chains

Markov chain Monte Carlo is a method to sample a given multivariate distribution π^* by constructing a suitable Markov chain with the property that its limiting, invariant distribution, is the target distribution π^* . In most problems of interest, the distribution π^* is absolutely continuous and, as a result, the theory of MCMC methods is based on that of Markov chains on continuous state spaces outlined, for example, in Nummelin (1984) and Meyn and Tweedie (1993). Tierney (1994) is the fundamental reference for drawing the connections between this elaborate Markov chain theory and MCMC methods. Basically, the goal of the analysis is to specify conditions under which the constructed Markov chain converges to the invariant distribution, and conditions under which sample path averages based on the output of the Markov chain satisfy a law of large numbers and a central limit theorem.

Definitions and Results

3.2.1

A Markov chain is a collection of random variables (or vectors) $\Phi = \{\Phi_i : i \in T\}$ where $T = \{0, 1, 2, \dots\}$. The evolution of the Markov chain on a space $\Omega \subseteq \mathfrak{R}^p$ is governed by the transition kernel

$$\begin{aligned} P(\mathbf{x}, A) &\equiv \Pr(\Phi_{i+1} \in A | \Phi_i = \mathbf{x}, \Phi_j, j < i) \\ &= \Pr(\Phi_{i+1} \in A | \Phi_i = \mathbf{x}), \quad \mathbf{x} \in \Omega, \quad A \subset \Omega, \end{aligned}$$

where the second line embodies the Markov property that the distribution of each succeeding state in the sequence, given the current and the past states, depends only on the current state.

Generally, the transition kernel in Markov chain simulations has both a continuous and discrete component. For some function $p(\mathbf{x}, \mathbf{y}) : \Omega \times \Omega \rightarrow \mathfrak{R}^+$, the kernel can be expressed as

$$P(\mathbf{x}, d\mathbf{y}) = p(\mathbf{x}, \mathbf{y})d\mathbf{y} + r(\mathbf{x})\delta_{\mathbf{x}}(d\mathbf{y}), \quad (3.2)$$

where $p(\mathbf{x}, \mathbf{x}) = 0$, $\delta_{\mathbf{x}}(d\mathbf{y}) = 1$ if $\mathbf{x} \in d\mathbf{y}$ and 0 otherwise, $r(\mathbf{x}) = 1 - \int_{\Omega} p(\mathbf{x}, \mathbf{y}) d\mathbf{y}$. This transition kernel specifies that transitions from \mathbf{x} to \mathbf{y} occur according to $p(\mathbf{x}, \mathbf{y})$ and transitions from \mathbf{x} to \mathbf{x} occur with probability $r(\mathbf{x})$.

The transition kernel is thus the distribution of Φ_{i+1} given that $\Phi_i = \mathbf{x}$. The n th step ahead transition kernel is given by

$$P^{(n)}(\mathbf{x}, A) = \int_{\Omega} P(\mathbf{x}, d\mathbf{y}) P^{(n-1)}(\mathbf{y}, A),$$

where $P^{(1)}(\mathbf{x}, d\mathbf{y}) = P(\mathbf{x}, d\mathbf{y})$ and

$$P(\mathbf{x}, A) = \int_A P(\mathbf{x}, d\mathbf{y}). \quad (3.3)$$

The goal is to find conditions under which the n th iterate of the transition kernel converges to the invariant distribution π^* as $n \rightarrow \infty$. The invariant distribution is one that satisfies

$$\pi^*(d\mathbf{y}) = \int_{\Omega} P(\mathbf{x}, d\mathbf{y})\pi(\mathbf{x}) d\mathbf{x}, \quad (3.4)$$

where π is the density of π^* with respect to the Lebesgue measure. The invariance condition states that if Φ_i is distributed according to π^* , then all subsequent elements of the chain are also distributed as π^* . Markov chain samplers are invariant by construction and therefore the existence of the invariant distribution does not have to be checked.

A Markov chain is reversible if the function $p(\mathbf{x}, \mathbf{y})$ in (3.2) satisfies

$$f(\mathbf{x})p(\mathbf{x}, \mathbf{y}) = f(\mathbf{y})p(\mathbf{y}, \mathbf{x}), \quad (3.5)$$

for a density $f(\cdot)$. If this condition holds, it can be shown that $f(\cdot) = \pi(\cdot)$ and has π^* as an invariant distribution (Tierney, 1994). To verify this we evaluate the right hand side of (3.4):

$$\begin{aligned} \int P(\mathbf{x}, A)\pi(\mathbf{x}) d\mathbf{x} &= \int \left\{ \int_A p(\mathbf{x}, \mathbf{y}) d\mathbf{y} \right\} \pi(\mathbf{x}) d\mathbf{x} + \int r(\mathbf{x})\delta_{\mathbf{x}}(A)\pi(\mathbf{x}) d\mathbf{x} \\ &= \int_A \left\{ \int p(\mathbf{x}, \mathbf{y})\pi(\mathbf{x}) d\mathbf{x} \right\} d\mathbf{y} + \int_A r(\mathbf{x})\pi(\mathbf{x}) d\mathbf{x} \\ &= \int_A \left\{ \int p(\mathbf{y}, \mathbf{x})\pi(\mathbf{y}) d\mathbf{x} \right\} d\mathbf{y} + \int_A r(\mathbf{x})\pi(\mathbf{x}) d\mathbf{x} \\ &= \int_A (1 - r(\mathbf{y}))\pi(\mathbf{y}) d\mathbf{y} + \int_A r(\mathbf{x})\pi(\mathbf{x}) d\mathbf{x} \\ &= \int_A \pi(\mathbf{y}) d\mathbf{y}. \end{aligned} \quad (3.6)$$

A minimal requirement on the Markov chain for it to satisfy a law of large numbers is the requirement of π^* -irreducibility. This means that the chain is able to visit all sets with positive probability under π^* from any starting point in Ω . Formally, a Markov chain is said to be π^* -irreducible if for every $\mathbf{x} \in \Omega$,

$$\pi^*(A) > 0 \Rightarrow P(\Phi_i \in A | \Phi_0 = \mathbf{x}) > 0$$

for some $i \geq 1$. If the space Ω is connected and the function $p(\mathbf{x}, \mathbf{y})$ is positive and continuous, then the Markov chain with transition kernel given by (3.3) and invariant distribution π^* is π^* -irreducible.

Another important property of a chain is aperiodicity, which ensures that the chain does not cycle through a finite number of sets. A Markov chain is aperiodic if there exists no partition of $\Omega = (D_0, D_1, \dots, D_{p-1})$ for some $p \geq 2$ such that $P(\Phi^i \in D_{i \bmod p} | \Phi_0 \in D_0) = 1$ for all i .

These definitions allow us to state the following results from Tierney (1994) which form the basis for Markov chain Monte Carlo methods. The first of these results gives conditions under which a strong law of large numbers holds and the second gives conditions under which the probability density of the M th iterate of the Markov chain converges to its unique, invariant density.

Theorem 1 Suppose $\{\Phi_i\}$ is a π^* -irreducible Markov chain with transition kernel $P(\cdot, \cdot)$ and invariant distribution π^* , then π^* is the unique invariant distribution of $P(\cdot, \cdot)$ and for all π^* -integrable real-valued functions h ,

1

$$\frac{1}{M} \sum_{i=1}^M h(\Phi_i) \rightarrow \int h(x)\pi(x)dx \quad \text{as } M \rightarrow \infty, \text{ a.s.}$$

Theorem 2 Suppose $\{\Phi_i\}$ is a π^* -irreducible, aperiodic Markov chain with transition kernel $P(\cdot, \cdot)$ and invariant distribution π^* . Then for π^* -almost every $x \in \Omega$, and all sets A

2

$$\| P^M(x, A) - \pi^*(A) \| \rightarrow 0 \quad \text{as } M \rightarrow \infty,$$

where $\| \cdot \|$ denotes the total variation distance.

A further strengthening of the conditions is required to obtain a central limit theorem for sample-path averages. A key requirement is that of an ergodic chain, i.e., chains that are irreducible, aperiodic and positive Harris-recurrent (for a definition of the latter, see Tierney (1994)). In addition, one needs the notion of geometric ergodicity. An ergodic Markov chain with invariant distribution π^* is a geometrically ergodic if there exists a non-negative real-valued function (bounded in expectation under π^*) and a positive constant $r < 1$ such that

$$\| P^n(x, A) - \pi^*(A) \| \leq C(x)r^n$$

for all x and all n and sets A . Chan and Ledolter (1995) show that if the Markov chain is ergodic, has invariant distribution π^* , and is geometrically ergodic, then for all L^2 measurable functions h , taken to be scalar-valued for simplicity, and any initial distribution, the distribution of $\sqrt{M}(\hat{h}_M - Eh)$ converges weakly to a normal distribution with mean zero and variance $\sigma_h^2 \geq 0$, where

$$\hat{h}_M = \frac{1}{M} \sum_{i=1}^M h(\Phi_i)$$

$$Eh = \int h(\Phi)\pi(\Phi)d\Phi$$

and

$$\sigma_h^2 = \text{Var } h(\Phi_0) + 2 \sum_{k=1}^{\infty} \text{Cov} [h(\Phi_0), h(\Phi_k)]. \quad (3.7)$$

Computation of Numerical Accuracy and Inefficiency Factor

The square root of σ_h^2 is the numerical standard error of \hat{h}_M . To describe estimators of σ_h^2 that are consistent in M , let $Z_i = h(\Phi_i)$ ($i \leq M$). Then, due to the fact that $\{Z_i\}$ is a dependent sequence

$$\begin{aligned} \text{Var}(\hat{h}_M) &= M^{-2} \sum_{j,k} \text{Cov}(Z_j, Z_k) \\ &= s^2 M^{-2} \sum_{j,k=1}^M \rho_{|j-k|} \\ &= s^2 M^{-1} \left\{ 1 + 2 \sum_{s=1}^M \left(1 - \frac{s}{M}\right) \rho_s \right\}, \end{aligned}$$

where s^2 is the sample variance of $\{Z_i\}$ and ρ_s is the estimated autocorrelation at lag s (see Ripley, 1987, Chap. 6). If $\rho_s > 0$ for each s , then this variance is larger than s^2/M which is the variance under independence. Another estimate of the variance can be found by consistently estimating the spectral density f of $\{Z_i\}$ at frequency zero and using the fact that $\text{Var}(\hat{h}_M) = \tau^2/M$, where $\tau^2 = 2\pi f(0)$. Finally, a traditional approach to finding the variance is by the method of batch means. In this approach, the data (Z_1, \dots, Z_M) is divided into k batches of length m with means $B_i = m^{-1}[Z_{(i-1)m+1} + \dots + Z_{im}]$ and the variance of \hat{h}_M estimated as

$$\text{Var}(\hat{h}_M) = \frac{1}{k(k-1)} \sum_{i=1}^k (B_i - \bar{B})^2, \quad (3.8)$$

where the batch size m is chosen to ensure that the first order serial correlation of the batch means is less than 0.05.

Given the numerical variance it is common to calculate the inefficiency factor, which is also called the autocorrelation time, defined as

$$\kappa_h = \frac{\text{Var}(\hat{h}_M)}{s^2/M}. \quad (3.9)$$

This quantity is interpreted as the ratio of the numerical variance of \hat{h}_M to the variance of \hat{h}_M based on independent draws, and its inverse is the relative numerical efficiency defined in Geweke (1992). Because independence sampling produces an autocorrelation time that is theoretically equal to one and Markov chain sampling produces autocorrelation times that are bigger than one, the inefficiency factor serves to quantify the relative efficiency loss in the computation of \hat{h}_M from correlated versus independent samples.

Metropolis–Hastings Algorithm

This powerful algorithm provides a general approach for producing a correlated sequence of draws from the target density that may be difficult to sample by a classical independence method. The goal is to simulate the d -dimensional distribution $\pi^*(\boldsymbol{\psi})$, $\boldsymbol{\psi} \in \Psi \subseteq \mathfrak{R}^d$ that has density $\pi(\boldsymbol{\psi})$ with respect to some dominating measure. To define the algorithm, let $q(\boldsymbol{\psi}, \boldsymbol{\psi}')$ denote a source density for a candidate draw $\boldsymbol{\psi}'$ given the current value $\boldsymbol{\psi}$ in the sampled sequence. The density $q(\boldsymbol{\psi}, \boldsymbol{\psi}')$ is referred to as the proposal or candidate generating density. Then, the M–H algorithm is defined by two steps: a first step in which a proposal value is drawn from the candidate generating density and a second step in which the proposal value is accepted as the next iterate in the Markov chain according to the probability $\alpha(\boldsymbol{\psi}, \boldsymbol{\psi}')$, where

$$\alpha(\boldsymbol{\psi}, \boldsymbol{\psi}') = \begin{cases} \min \left[\frac{\pi(\boldsymbol{\psi}')q(\boldsymbol{\psi}', \boldsymbol{\psi})}{\pi(\boldsymbol{\psi})q(\boldsymbol{\psi}, \boldsymbol{\psi}')}, 1 \right] & \text{if } \pi(\boldsymbol{\psi})q(\boldsymbol{\psi}, \boldsymbol{\psi}') > 0 ; \\ 1 & \text{otherwise .} \end{cases} \quad (3.10)$$

If the proposal value is rejected, then the next sampled value is taken to be the current value. In algorithmic form, the simulated values are obtained by the following recursive procedure.

Algorithm 1: *Metropolis–Hastings*

1. Specify an initial value $\boldsymbol{\psi}^{(0)}$;
2. Repeat for $j = 1, 2, \dots, M$.
 - a) Propose

$$\boldsymbol{\psi}' \sim q(\boldsymbol{\psi}^{(j)}, \cdot)$$

- b) Let

$$\boldsymbol{\psi}^{(j+1)} = \begin{cases} \boldsymbol{\psi}' & \text{if } \text{Unif}(0, 1) \leq \alpha(\boldsymbol{\psi}^{(j)}, \boldsymbol{\psi}') ; \\ \boldsymbol{\psi}^{(j)} & \text{otherwise .} \end{cases}$$

3. Return the values $\{\boldsymbol{\psi}^{(1)}, \boldsymbol{\psi}^{(2)}, \dots, \boldsymbol{\psi}^{(M)}\}$.

Typically, a certain number of values (say n_0) at the start of this sequence are discarded after which the chain is assumed to have converged to its invariant distribution and the subsequent draws are taken as approximate variates from π . Because theoretical calculation of the burn-in is not easy it is important that the proposal density is chosen to ensure that the chain makes large moves through the support of the invariant distribution without staying at one place for many iterations. Generally, the empirical behavior of the M–H output is monitored by

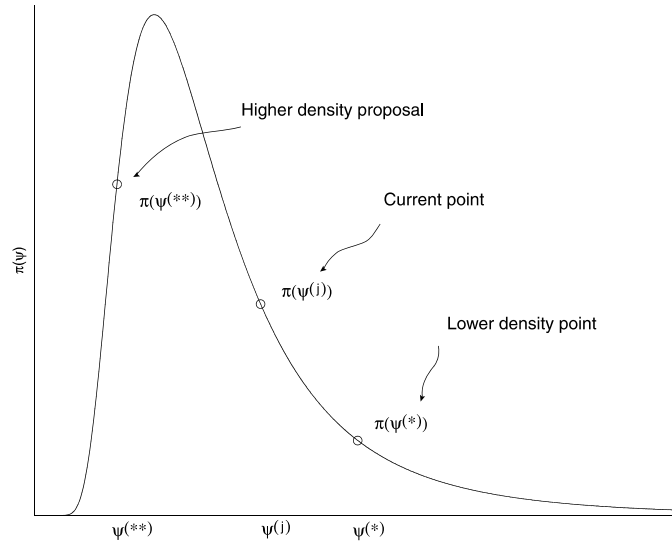


Figure 3.1. Original Metropolis algorithm: higher density proposal is accepted with probability one and the lower density proposal with probability α

the autocorrelation time of each component of $\boldsymbol{\psi}$ and by the acceptance rate, which is the proportion of times a move is made as the sampling proceeds.

One should observe that the target density appears as a ratio in the probability $\alpha(\boldsymbol{\psi}, \boldsymbol{\psi}')$ and therefore the algorithm can be implemented without knowledge of the normalizing constant of $\pi(\cdot)$. Furthermore, if the candidate-generating density is symmetric, i.e. $q(\boldsymbol{\psi}, \boldsymbol{\psi}') = q(\boldsymbol{\psi}', \boldsymbol{\psi})$, the acceptance probability only contains the ratio $\pi(\boldsymbol{\psi}')/\pi(\boldsymbol{\psi})$; hence, if $\pi(\boldsymbol{\psi}') \geq \pi(\boldsymbol{\psi})$, the chain moves to $\boldsymbol{\psi}'$, otherwise it moves with probability given by $\pi(\boldsymbol{\psi}')/\pi(\boldsymbol{\psi})$. The latter is the algorithm originally proposed by Metropolis et al. (1953). This version of the algorithm is illustrated in Fig. 3.1.

Different proposal densities give rise to specific versions of the M–H algorithm, each with the correct invariant distribution π . One family of candidate-generating densities is given by $q(\boldsymbol{\psi}, \boldsymbol{\psi}') = q(\boldsymbol{\psi}' - \boldsymbol{\psi})$. The candidate $\boldsymbol{\psi}'$ is thus drawn according to the process $\boldsymbol{\psi}' = \boldsymbol{\psi} + \mathbf{z}$, where \mathbf{z} follows the distribution q . Since the candidate is equal to the current value plus noise, this case is called a random walk M–H chain. Possible choices for q include the multivariate normal density and the multivariate- t . The random walk M–H chain is perhaps the simplest version of the M–H algorithm (and was the one used by Metropolis et al., 1953) and popular in applications. One has to be careful, however, in setting the variance of \mathbf{z} ; if it is too large it is possible that the chain may remain stuck at a particular value for many iterations while if it is too small the chain will tend to make small moves and move inefficiently through the support of the target distribution. In both cases the generated draws that will be highly serially correlated. Note that

when q is symmetric, $q(z) = q(-z)$ and the probability of move only contains the ratio $\pi(\boldsymbol{\psi}')/\pi(\boldsymbol{\psi})$. As mentioned earlier, the same reduction occurs if $q(\boldsymbol{\psi}, \boldsymbol{\psi}') = q(\boldsymbol{\psi}', \boldsymbol{\psi})$.

Hastings (1970) considers a second family of candidate-generating densities that are given by the form $q(\boldsymbol{\psi}, \boldsymbol{\psi}') = q(\boldsymbol{y})$. Tierney (1994) refers to this as an independence M–H chain because, in contrast to the random walk chain, the candidates are drawn independently of the current location $\boldsymbol{\psi}$. In this case, the probability of move becomes

$$\alpha(\boldsymbol{\psi}, \boldsymbol{\psi}') = \min \left\{ \frac{w(\boldsymbol{\psi}')}{w(\boldsymbol{\psi})}, 1 \right\} ;$$

where $w(\boldsymbol{\psi}) = \pi(\boldsymbol{\psi})/q(\boldsymbol{\psi})$ is the ratio of the target and proposal densities. For this method to work and not get stuck in the tails of π , it is important that the proposal density have thicker tails than π . A similar requirement is placed on the importance sampling function in the method of importance sampling (Geweke, 1989). In fact, Mengersen and Tweedie (1996) show that if $w(\boldsymbol{\psi})$ is uniformly bounded then the resulting Markov chain is ergodic.

Chib and Greenberg (1994, 1995) discuss a way of formulating proposal densities in the context of time series autoregressive-moving average models that has a bearing on the choice of proposal density for the independence M–H chain. They suggest matching the proposal density to the target at the mode by a multivariate normal or multivariate- t distribution with location given by the mode of the target and the dispersion given by inverse of the Hessian evaluated at the mode. Specifically, the parameters of the proposal density are taken to be

$$\begin{aligned} \boldsymbol{m} &= \arg \max \log \pi(\boldsymbol{\psi}) \quad \text{and} \\ \boldsymbol{V} &= \tau \left\{ -\frac{\partial^2 \log \pi(\boldsymbol{\psi})}{\partial \boldsymbol{\psi} \partial \boldsymbol{\psi}'} \right\}_{\boldsymbol{\psi}=\hat{\boldsymbol{\psi}}}^{-1}, \end{aligned} \tag{3.11}$$

where τ is a tuning parameter that is adjusted to control the acceptance rate. The proposal density is then specified as $q(\boldsymbol{\psi}') = f(\boldsymbol{\psi}'|\boldsymbol{m}, \boldsymbol{V})$, where f is some multivariate density. This may be called a tailored M–H chain.

Another way to generate proposal values is through a Markov chain version of the accept-reject method. In this version, due to Tierney (1994), and considered in detail by Chib and Greenberg (1995), a pseudo accept-reject step is used to generate candidates for an M–H algorithm. Suppose $c > 0$ is a known constant and $h(\boldsymbol{\psi})$ a source density. Let $C = \{\boldsymbol{\psi} : \pi(\boldsymbol{\psi}) \leq ch(\boldsymbol{\psi})\}$ denote the set of value for which $ch(\boldsymbol{\psi})$ dominates the target density and assume that this set has high probability under π^* . Given $\boldsymbol{\psi}^{(n)} = \boldsymbol{\psi}$, the next value $\boldsymbol{\psi}^{(n+1)}$ is obtained as follows: First, a candidate value $\boldsymbol{\psi}'$ is obtained, independent of the current value $\boldsymbol{\psi}$, by applying the accept-reject algorithm with $ch(\cdot)$ as the “pseudo dominating” density. The candidates $\boldsymbol{\psi}'$ that are produced under this scheme have density $q(\boldsymbol{\psi}') \propto \min\{\pi(\boldsymbol{\psi}'), ch(\boldsymbol{\psi}')\}$.

If we let $w(\boldsymbol{\psi}) = c^{-1}\pi(\boldsymbol{\psi})/h(\boldsymbol{\psi})$ then it can be shown that the M–H probability of move is given by

$$\alpha(\boldsymbol{\psi}, \boldsymbol{\psi}') = \begin{cases} 1 & \text{if } \boldsymbol{\psi} \in C \\ 1/w(\boldsymbol{\psi}) & \text{if } \boldsymbol{\psi} \notin C, \boldsymbol{\psi}' \in C \\ \min\{w(\boldsymbol{\psi}')/w(\boldsymbol{\psi}), 1\} & \text{if } \boldsymbol{\psi} \notin C, \boldsymbol{\psi}' \notin C \end{cases} \quad (3.12)$$

3.3.1 Convergence Results

In the M–H algorithm the transition kernel of the chain is given by

$$P(\boldsymbol{\psi}, d\boldsymbol{\psi}') = q(\boldsymbol{\psi}, \boldsymbol{\psi}')\alpha(\boldsymbol{\psi}, \boldsymbol{\psi}') d\boldsymbol{\psi}' + r(\boldsymbol{\psi})\delta_{\boldsymbol{\psi}}(d\boldsymbol{\psi}'), \quad (3.13)$$

where $\delta_{\boldsymbol{\psi}}(d\boldsymbol{\psi}') = 1$ if $\boldsymbol{\psi} \in d\boldsymbol{\psi}'$ and 0 otherwise and

$$r(\boldsymbol{\psi}) = 1 - \int_{\Omega} q(\boldsymbol{\psi}, \boldsymbol{\psi}')\alpha(\boldsymbol{\psi}, \boldsymbol{\psi}') d\boldsymbol{\psi}' .$$

Thus, transitions from $\boldsymbol{\psi}$ to $\boldsymbol{\psi}'$ ($\boldsymbol{\psi}' \neq \boldsymbol{\psi}$) are made according to the density

$$p(\boldsymbol{\psi}, \boldsymbol{\psi}') \equiv q(\boldsymbol{\psi}, \boldsymbol{\psi}')\alpha(\boldsymbol{\psi}, \boldsymbol{\psi}'), \quad \boldsymbol{\psi} \neq \boldsymbol{\psi}'$$

while transitions from $\boldsymbol{\psi}$ to $\boldsymbol{\psi}$ occur with probability $r(\boldsymbol{\psi})$. In other words, the density function implied by this transition kernel is of mixed type,

$$K(\boldsymbol{\psi}, \boldsymbol{\psi}') = q(\boldsymbol{\psi}, \boldsymbol{\psi}')\alpha(\boldsymbol{\psi}, \boldsymbol{\psi}') + r(\boldsymbol{\psi})\delta_{\boldsymbol{\psi}}(\boldsymbol{\psi}'), \quad (3.14)$$

having both a continuous and discrete component, where now, with change of notation, $\delta_{\boldsymbol{\psi}}(\boldsymbol{\psi}')$ is the Dirac delta function defined as $\delta_{\boldsymbol{\psi}}(\boldsymbol{\psi}') = 0$ for $\boldsymbol{\psi}' \neq \boldsymbol{\psi}$ and $\int_{\Omega} \delta_{\boldsymbol{\psi}}(\boldsymbol{\psi}') d\boldsymbol{\psi}' = 1$.

Chib and Greenberg (1995) provide a way to derive and interpret the probability of move $\alpha(\boldsymbol{\psi}, \boldsymbol{\psi}')$. Consider the proposal density $q(\boldsymbol{\psi}, \boldsymbol{\psi}')$. This proposal density q is not likely to be reversible for π (if it were then we would be done and M–H sampling would not be necessary). Without loss of generality, suppose that $\pi(\boldsymbol{\psi})q(\boldsymbol{\psi}, \boldsymbol{\psi}') > \pi(\boldsymbol{\psi}')q(\boldsymbol{\psi}', \boldsymbol{\psi})$ implying that the rate of transitions from $\boldsymbol{\psi}$ to $\boldsymbol{\psi}'$ exceed those in the reverse direction. To reduce the transitions from $\boldsymbol{\psi}$ to $\boldsymbol{\psi}'$ one can introduce a function $0 \leq \alpha(\boldsymbol{\psi}, \boldsymbol{\psi}') \leq 1$ such that $\pi(\boldsymbol{\psi})q(\boldsymbol{\psi}, \boldsymbol{\psi}')\alpha(\boldsymbol{\psi}, \boldsymbol{\psi}') = \pi(\boldsymbol{\psi}')q(\boldsymbol{\psi}', \boldsymbol{\psi})$. Solving for $\alpha(\boldsymbol{\psi}, \boldsymbol{\psi}')$ yields the probability of move in the M–H algorithm. This calculation reveals the important point that the function $p(\boldsymbol{\psi}, \boldsymbol{\psi}') = q(\boldsymbol{\psi}, \boldsymbol{\psi}')\alpha(\boldsymbol{\psi}, \boldsymbol{\psi}')$ is reversible by construction, i.e., it satisfies the condition

$$q(\boldsymbol{\psi}, \boldsymbol{\psi}')\alpha(\boldsymbol{\psi}, \boldsymbol{\psi}')\pi(\boldsymbol{\psi}) = q(\boldsymbol{\psi}', \boldsymbol{\psi})\alpha(\boldsymbol{\psi}', \boldsymbol{\psi})\pi(\boldsymbol{\psi}') . \quad (3.15)$$

It immediately follows, therefore, from the argument in (3.6) that the M–H kernel has $\pi(\boldsymbol{\psi})$ as its invariant density.

It is not difficult to provide conditions under which the Markov chain generated by the M–H algorithm satisfies the conditions of Propositions 1–2. The conditions

of Proposition 1 are satisfied by the M–H chain if $q(\boldsymbol{\psi}, \boldsymbol{\psi}')$ is positive for $(\boldsymbol{\psi}, \boldsymbol{\psi}')$ and continuous and the set $\boldsymbol{\psi}$ is connected. In addition, the conditions of Proposition 2 are satisfied if q is not reversible (which is the usual situation) which leads to a chain that is aperiodic. Conditions for ergodicity, required for use of the central limit theorem, are satisfied if in addition π is bounded. Other similar conditions are provided by Robert and Casella (1999).

Example

3.3.2

To illustrate the M–H algorithm, consider the binary response data in Table 3.1, taken from Fahrmeir and Tutz (1997), on the occurrence or non-occurrence of infection following birth by caesarean section. The response variable y is one if the caesarean birth resulted in an infection, and zero if not. There are three covariates: x_1 , an indicator of whether the caesarean was non-planned; x_2 , an indicator of whether risk factors were present at the time of birth and x_3 , an indicator of whether antibiotics were given as a prophylaxis. The data in the table contains information from 251 births. Under the column of the response, an entry such as 11/87 means that there were 98 deliveries with covariates (1, 1, 1) of whom 11 developed an infection and 87 did not.

Table 3.1. Caesarean infection data

Y (1/0)	x_1	x_2	x_3
11/87	1	1	1
1/17	0	1	1
0/2	0	0	1
23/3	1	1	0
28/30	0	1	0
0/9	1	0	0
8/32	0	0	0

Suppose that the probability of infection for the i th birth ($i \leq 251$) is

$$\Pr(y_i = 1 | \mathbf{x}_i, \boldsymbol{\beta}) = \Phi(\mathbf{x}_i^\top \boldsymbol{\beta}), \tag{3.16}$$

$$\boldsymbol{\beta} \sim N_4(0, 5\mathbf{I}_4), \tag{3.17}$$

where $\mathbf{x}_i = (1, x_{i1}, x_{i2}, x_{i3})^\top$ is the covariate vector, $\boldsymbol{\beta} = (\beta_0, \beta_1, \beta_2, \beta_3)$ is the vector of unknown coefficients, Φ is the cdf of the standard normal random variable and \mathbf{I}_4 is the four-dimensional identity matrix. The target posterior density, under the assumption that the outcomes $\mathbf{y} = (y_1, y_2, \dots, y_{251})$ are conditionally independent, is

$$\pi(\boldsymbol{\beta} | \mathbf{y}) \propto \pi(\boldsymbol{\beta}) \prod_{i=1}^{251} \Phi(\mathbf{x}_i^\top \boldsymbol{\beta})^{y_i} \{1 - \Phi(\mathbf{x}_i^\top \boldsymbol{\beta})\}^{(1-y_i)},$$

where $\pi(\boldsymbol{\beta})$ is the density of the $N(0, 10\mathbf{I}_4)$ distribution.

Random Walk Proposal Density

To define the proposal density, let

$$\hat{\boldsymbol{\beta}} = (-1.093022 \quad 0.607643 \quad 1.197543 \quad -1.904739)^\top$$

be the MLE found using the Newton–Raphson algorithm and let

$$\mathbf{V} = \begin{pmatrix} 0.040745 & -0.007038 & -0.039399 & 0.004829 \\ & 0.073101 & -0.006940 & -0.050162 \\ & & 0.062292 & -0.016803 \\ & & & 0.080788 \end{pmatrix}$$

be the symmetric matrix obtained by inverting the negative of the Hessian matrix (the matrix of second derivatives) of the log-likelihood function evaluated at $\hat{\boldsymbol{\beta}}$. Now generate the proposal values by the random walk:

$$\begin{aligned} \boldsymbol{\beta} &= \boldsymbol{\beta}^{(j-1)} + \boldsymbol{\varepsilon}^{(j)} \\ \boldsymbol{\varepsilon}^{(j)} &\sim N_4(\mathbf{0}, \mathbf{V}), \end{aligned} \quad (3.18)$$

which leads to the original Metropolis method. From a run of 5000 iterations of the algorithm beyond a burn-in of a 100 iterations we get the prior-posterior summary that is reported in Table 3.2, which contains the first two moments of the prior and posterior and the 2.5th (lower) and 97.5th (upper) percentiles of the marginal densities of $\boldsymbol{\beta}$.

Table 3.2. Caesarean data: Prior-posterior summary based on 5000 draws (beyond a burn-in of 100 cycles) from the random-walk M–H algorithm

	Prior		Posterior			
	Mean	Std dev	Mean	Std dev	Lower	Upper
β_0	0.000	3.162	-1.110	0.224	-1.553	-0.677
β_1	0.000	3.162	0.612	0.254	0.116	1.127
β_2	0.000	3.162	1.198	0.263	0.689	1.725
β_3	0.000	3.162	-1.901	0.275	-2.477	-1.354

As expected, both the first and second covariates increase the probability of infection while the third covariate (the antibiotics prophylaxis) reduces the probability of infection.

To get an idea of the form of the posterior density we plot in Fig. 3.1 the four marginal posterior densities. The density plots are obtained by smoothing the histogram of the simulated values with a Gaussian kernel. In the same plot we also report the autocorrelation functions (correlation against lag) for each of the sampled parameter values. The autocorrelation plots provide information of the

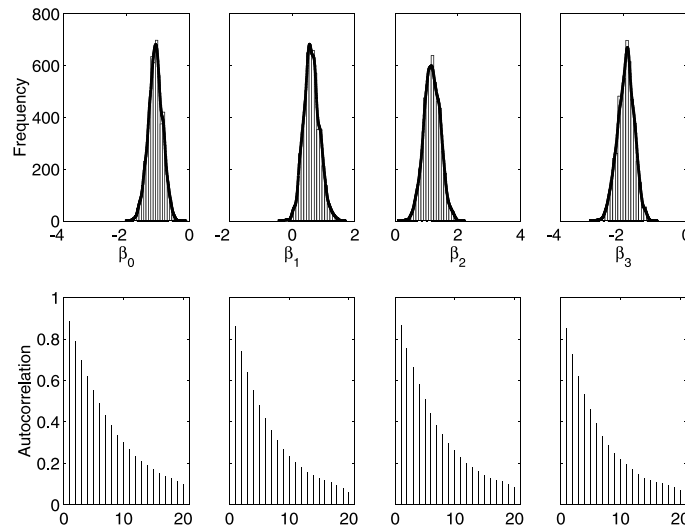


Figure 3.2. Caesarean data with random-walk M-H algorithm: Marginal posterior densities (*top panel*) and autocorrelation plot (*bottom panel*)

extent of serial dependence in the sampled values. Here we see that the serial correlations start out high but decline to almost zero by lag twenty.

Tailored Proposal Density

To see the difference in results, the M-H algorithm is next implemented with a tailored proposal density. In this scheme one utilizes both $\hat{\beta}$ and V that were defined above. We let the proposal density be $f_T(\beta|\hat{\beta}, V, 15)$, a multivariate- t density with fifteen degrees of freedom. This proposal density is similar to the random-walk proposal except that the distribution is centered at the fixed point $\hat{\beta}$. The prior-posterior summary based on 5000 draws of the M-H algorithm with this proposal density is given in Table 3.3. We see that the marginal posterior moments are similar to those in Table 3.1. The marginal posterior densities are reported in the top panel of Fig. 3.2. These are virtually identical to those computed using

Table 3.3. Caesarean data: Prior-posterior summary based on 5000 draws (beyond a burn-in of 100 cycles) from the tailored M-H algorithm

	Prior		Posterior			
	Mean	Std dev	Mean	Std dev	Lower	Upper
β_0	0.000	3.162	-1.080	0.220	-1.526	-0.670
β_1	0.000	3.162	0.593	0.249	0.116	1.095
β_2	0.000	3.162	1.181	0.254	0.680	1.694
β_3	0.000	3.162	-1.889	0.266	-2.421	-1.385

the random-walk M–H algorithm. The most notable difference is in the serial correlation plots which decline much more quickly to zero indicating that the algorithm is mixing well. The same information is revealed by the inefficiency factors which are much closer to one than those from the previous algorithm.

The message from this analysis is that the two proposal densities produce similar results, with the differences appearing only in the autocorrelation plots (and inefficiency factors) of the sampled draws.

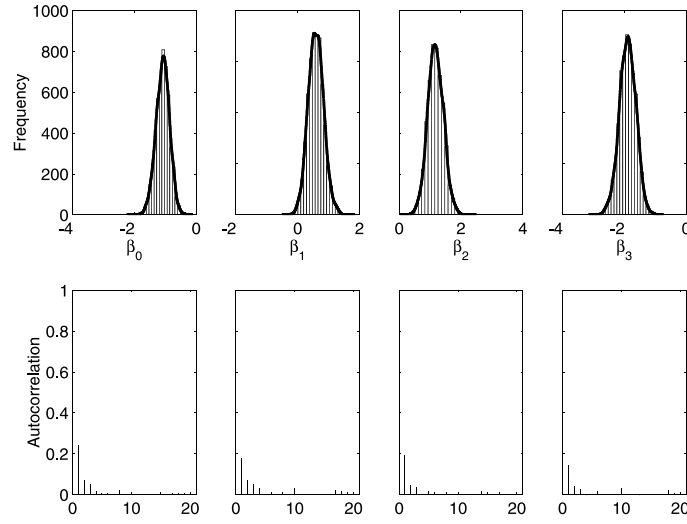


Figure 3.3. Caesarian data with tailored M–H algorithm: Marginal posterior densities (*top panel*) and autocorrelation plot (*bottom panel*)

3.3.3 Multiple-Block M–H Algorithm

In applications when the dimension of $\boldsymbol{\psi}$ is large, it can be difficult to construct a single block M–H algorithm that converges rapidly to the target density. In such cases, it is helpful to break up the variate space into smaller blocks and to then construct a Markov chain with these smaller blocks. Suppose, for illustration, that $\boldsymbol{\psi}$ is split into two vector blocks $(\boldsymbol{\psi}_1, \boldsymbol{\psi}_2)$. For example, in a regression model, one block may consist of the regression coefficients and the other block may consist of the error variance. Next, for each block, let

$$q_1(\boldsymbol{\psi}_1, \boldsymbol{\psi}'_1 | \boldsymbol{\psi}_2); \quad q_2(\boldsymbol{\psi}_2, \boldsymbol{\psi}'_2 | \boldsymbol{\psi}_1),$$

denote the corresponding proposal density. Here each proposal density q_k is allowed to depend on the data and the current value of the remaining block. Also define (by analogy with the single-block case)

$$\alpha(\boldsymbol{\psi}_1, \boldsymbol{\psi}'_1 | \boldsymbol{\psi}_2) = \min \left\{ 1, \frac{\pi(\boldsymbol{\psi}'_1 | \boldsymbol{\psi}_2) q_1(\boldsymbol{\psi}_1, \boldsymbol{\psi}'_1 | \boldsymbol{\psi}_2)}{\pi(\boldsymbol{\psi}_1 | \boldsymbol{\psi}_2) q_1(\boldsymbol{\psi}_1, \boldsymbol{\psi}'_1 | \boldsymbol{\psi}_2)} \right\}, \quad (3.19)$$

and

$$\alpha(\boldsymbol{\psi}_2, \boldsymbol{\psi}'_2 | \boldsymbol{y}, \boldsymbol{\psi}_1) = \min \left\{ 1, \frac{\pi(\boldsymbol{\psi}'_2 | \boldsymbol{\psi}_1) q_2(\boldsymbol{\psi}'_2, \boldsymbol{\psi}_2 | \boldsymbol{\psi}_1)}{\pi(\boldsymbol{\psi}_2 | \boldsymbol{\psi}_1) q_2(\boldsymbol{\psi}_2, \boldsymbol{\psi}'_2 | \boldsymbol{\psi}_1)} \right\}, \quad (3.20)$$

as the probability of move for block $\boldsymbol{\psi}_k$ ($k = 1, 2$) conditioned on the other block. The conditional densities

$$\pi(\boldsymbol{\psi}_1 | \boldsymbol{\psi}_2) \quad \text{and} \quad \pi(\boldsymbol{\psi}_2 | \boldsymbol{\psi}_1)$$

that appear in these functions are called the *full conditional densities*. By Bayes theorem each is proportional to the joint density. For example,

$$\pi(\boldsymbol{\psi}_1 | \boldsymbol{\psi}_2) \propto \pi(\boldsymbol{\psi}_1, \boldsymbol{\psi}_2),$$

and, therefore, the probabilities of move in (3.19) and (3.20) can be expressed equivalently in terms of the kernel of the joint posterior density $\pi(\boldsymbol{\psi}_1, \boldsymbol{\psi}_2)$ because the normalizing constant of the full conditional density (the norming constant in the latter expression) cancels in forming the ratio.

With these inputs, one sweep of the multiple-block M-H algorithm is completed by updating each block, say sequentially in fixed order, using a M-H step with the above probabilities of move, given the most current value of the other block.

Algorithm 2: *Multiple-Block Metropolis-Hastings*

1. Specify an initial value $\boldsymbol{\psi}^{(0)} = (\boldsymbol{\psi}_1^{(0)}, \boldsymbol{\psi}_2^{(0)})$:
2. Repeat for $j = 1, 2, \dots, n_0 + M$.
 - a) Repeat for $k = 1, 2$
 - I. Propose a value for the k th block, conditioned on the previous value of k th block, and the current value of the other block $\boldsymbol{\psi}_{-k}$:

$$\boldsymbol{\psi}'_k \sim q_k(\boldsymbol{\psi}_k^{(j-1)}, \cdot | \boldsymbol{\psi}_{-k}).$$

- II. Calculate the probability of move

$$\alpha_k(\boldsymbol{\psi}_k^{(j-1)}, \boldsymbol{\psi}'_k | \boldsymbol{y}, \boldsymbol{\psi}_{-k}) = \min \left\{ 1, \frac{\pi(\boldsymbol{\psi}'_k | \boldsymbol{\psi}_{-k}) q_k(\boldsymbol{\psi}'_k, \boldsymbol{\psi}_k^{(j-1)} | \boldsymbol{\psi}_{-k})}{h(\boldsymbol{\psi}_k^{(j-1)} | \boldsymbol{\psi}_{-k}) q_k(\boldsymbol{\psi}_k^{(j-1)}, \boldsymbol{\psi}'_k | \boldsymbol{\psi}_{-k})} \right\}.$$

- III. Update the k th block as

$$\boldsymbol{\psi}_k^{(j)} = \begin{cases} \boldsymbol{\psi}'_k & \text{with prob } \alpha_k(\boldsymbol{\psi}_k^{(j-1)}, \boldsymbol{\psi}'_k | \boldsymbol{\psi}_{-k}) \\ \boldsymbol{\psi}_k^{(j-1)} & \text{with prob } 1 - \alpha_k(\boldsymbol{\psi}_k^{(j-1)}, \boldsymbol{\psi}'_k | \boldsymbol{\psi}_{-k}) \end{cases}.$$

3. Return the values $\{\boldsymbol{\psi}^{(n_0+1)}, \boldsymbol{\psi}^{(n_0+2)}, \dots, \boldsymbol{\psi}^{(n_0+M)}\}$.

The extension of this method to more than two blocks is straightforward.

The transition kernel of the resulting Markov chain is given by the product of transition kernels

$$P(\boldsymbol{\psi}, d\boldsymbol{\psi}') = \prod_{k=1}^2 P_k(\boldsymbol{\psi}_k, d\boldsymbol{\psi}'_k | \boldsymbol{\psi}_{-k}) \quad (3.21)$$

This transition kernel is not reversible, as can be easily checked, because under fixed sequential updating of the blocks updating in the reverse order never occurs. The multiple-block M-H algorithm, however, satisfies the weaker condition of invariance. To show this, we utilize the fact that each sub-move satisfies local reversibility (Chib and Jeliazkov (2001)) and therefore the transition kernel $P_1(\boldsymbol{\psi}_1, d\boldsymbol{\psi}_1 | \boldsymbol{\psi}_2)$ has $\pi_{1|2}^*(\cdot | \boldsymbol{\psi}_2)$ as its local invariant distribution with density $\pi_{1|2}^*(\cdot | \boldsymbol{\psi}_2)$, i.e.,

$$\pi_{1|2}^*(d\boldsymbol{\psi}_1 | \boldsymbol{\psi}_2) = \int P_1(\boldsymbol{\psi}_1, d\boldsymbol{\psi}_1 | \boldsymbol{\psi}_2) \pi_{1|2}(\boldsymbol{\psi}_1 | \boldsymbol{\psi}_2) d\boldsymbol{\psi}_1 . \quad (3.22)$$

Similarly, the conditional transition kernel $P_2(\boldsymbol{\psi}_2, d\boldsymbol{\psi}_2 | \boldsymbol{\psi}_1)$ has $\pi_{2|1}^*(\cdot | \boldsymbol{\psi}_1)$ as its invariant distribution, for a given value of $\boldsymbol{\psi}_1$. Then, the kernel formed by multiplying the conditional kernels is invariant for $\pi^*(\cdot, \cdot)$:

$$\begin{aligned} & \iint P_1(\boldsymbol{\psi}_1, d\boldsymbol{\psi}'_1 | \boldsymbol{\psi}_2) P_2(\boldsymbol{\psi}_2, d\boldsymbol{\psi}'_2 | \boldsymbol{\psi}'_1) \pi(\boldsymbol{\psi}_1, \boldsymbol{\psi}_2) d\boldsymbol{\psi}_1 d\boldsymbol{\psi}_2 \\ &= \int P_2(\boldsymbol{\psi}_2, d\boldsymbol{\psi}'_2 | \boldsymbol{\psi}'_1) \left[\int P_1(\boldsymbol{\psi}_1, d\boldsymbol{\psi}'_1 | \boldsymbol{\psi}_2) \pi_{1|2}(\boldsymbol{\psi}_1 | \boldsymbol{\psi}_2) d\boldsymbol{\psi}_1 \right] \pi_2(\boldsymbol{\psi}_2) d\boldsymbol{\psi}_2 \\ &= \int P_2(\boldsymbol{\psi}_2, d\boldsymbol{\psi}'_2 | \boldsymbol{\psi}'_1) \pi_{1|2}^*(d\boldsymbol{\psi}'_1 | \boldsymbol{\psi}_2) \pi_2(\boldsymbol{\psi}_2) d\boldsymbol{\psi}_2 \\ &= \int P_2(\boldsymbol{\psi}_2, d\boldsymbol{\psi}'_2 | \boldsymbol{\psi}'_1) \frac{\pi_{2|1}(\boldsymbol{\psi}_2 | \boldsymbol{\psi}'_1) \pi_1^*(d\boldsymbol{\psi}'_1)}{\pi_2(\boldsymbol{\psi}_2)} \pi_2(\boldsymbol{\psi}_2) d\boldsymbol{\psi}_2 \\ &= \pi_1^*(d\boldsymbol{\psi}'_1) \int P_2(\boldsymbol{\psi}_2, d\boldsymbol{\psi}'_2 | \boldsymbol{\psi}'_1) \pi_{2|1}(\boldsymbol{\psi}_2 | \boldsymbol{\psi}'_1) d\boldsymbol{\psi}_2 \\ &= \pi_1^*(d\boldsymbol{\psi}'_1) \pi_{2|1}^*(d\boldsymbol{\psi}'_2 | \boldsymbol{\psi}'_1) \\ &= \pi^*(d\boldsymbol{\psi}'_1, d\boldsymbol{\psi}'_2) , \end{aligned}$$

where the third line follows from (3.22), the fourth from Bayes theorem, the sixth from assumed invariance of P_2 , and the last from the law of total probability.

The implication of this result is that it allows us to take draws in succession from each of the kernels, instead of having to run each to convergence for every value of the conditioning variable.

1

Remark 1 Versions of either random-walk or tailored proposal densities can be used in this algorithm, analogous to the single-block case. For example, Chib and Greenberg (1995) determine the proposal densities q_k by tailoring to $\pi(\boldsymbol{\psi}_k, \boldsymbol{\psi}_{-k})$ in which case the proposal density is not fixed but varies across iterations. An

important special case occurs if each proposal density is taken to be the full conditional density of that block. Specifically, if we set

$$q_1(\boldsymbol{\psi}_1^{(j-1)}, \boldsymbol{\psi}'_1 | \boldsymbol{\psi}_2) = \pi(\boldsymbol{\psi}'_1 | \boldsymbol{\psi}_2),$$

and

$$q_2(\boldsymbol{\psi}_2^{(j-1)}, \boldsymbol{\psi}'_2 | \boldsymbol{\psi}_1) = \pi(\boldsymbol{\psi}'_2 | \boldsymbol{\psi}_1),$$

then an interesting simplification occurs. The probability of move (for the first block) becomes

$$\begin{aligned} \alpha_1(\boldsymbol{\psi}_1^{(j-1)}, \boldsymbol{\psi}'_1 | \boldsymbol{\psi}_2) &= \min \left\{ 1, \frac{\pi(\boldsymbol{\psi}'_1 | \boldsymbol{\psi}_2) \pi(\boldsymbol{\psi}_1^{(j-1)} | \boldsymbol{\psi}_2)}{\pi(\boldsymbol{\psi}_1^{(j-1)} | \boldsymbol{\psi}_2) \pi(\boldsymbol{\psi}'_1 | \boldsymbol{\psi}_2)} \right\} \\ &= 1, \end{aligned}$$

and similarly for the second block, implying that if proposal values are drawn from their full conditional densities then the proposal values are accepted with probability one. This special case of the multiple-block M-H algorithm (in which *each* block is proposed using its full conditional distribution) is called the Gibbs sampling algorithm.

The Gibbs Sampling Algorithm

3.4

The Gibbs sampling algorithm is one of the simplest Markov chain Monte Carlo algorithms. It was introduced by Geman and Geman (1984) in the context of image processing and then discussed in the context of missing data problems by Tanner and Wong (1987). The paper by Gelfand and Smith (1990) helped to demonstrate the value of the Gibbs algorithm for a range of problems in Bayesian analysis.

The Algorithm

3.4.1

To define the Gibbs sampling algorithm, let the set of full conditional distributions be

$$\{\pi(\boldsymbol{\psi}_1 | \boldsymbol{\psi}_2, \dots, \boldsymbol{\psi}_p); \pi(\boldsymbol{\psi}_2 | \boldsymbol{\psi}_1, \boldsymbol{\psi}_3, \dots, \boldsymbol{\psi}_p); \dots, \pi(\boldsymbol{\psi}_p | \boldsymbol{\psi}_1, \dots, \boldsymbol{\psi}_{d-1})\}.$$

Now one cycle of the Gibbs sampling algorithm is completed by simulating $\{\boldsymbol{\psi}_k\}_{k=1}^p$ from these distributions, recursively refreshing the conditioning variables. When $d = 2$ one obtains the two block Gibbs sampler that appears in Tanner and Wong (1987). The Gibbs sampler in which each block is revised in fixed order is defined as follows.

Algorithm 3: *Gibbs Sampling*

1. Specify an initial value $\boldsymbol{\psi}^{(0)} = (\boldsymbol{\psi}_1^{(0)}, \dots, \boldsymbol{\psi}_p^{(0)})$:
2. Repeat for $j = 1, 2, \dots, M$.
 - Generate $\boldsymbol{\psi}_1^{(j+1)}$ from $\pi(\boldsymbol{\psi}_1 | \boldsymbol{\psi}_2^{(j)}, \boldsymbol{\psi}_3^{(j)}, \dots, \boldsymbol{\psi}_p^{(j)})$
 - Generate $\boldsymbol{\psi}_2^{(j+1)}$ from $\pi(\boldsymbol{\psi}_2 | \boldsymbol{\psi}_1^{(j+1)}, \boldsymbol{\psi}_3^{(j)}, \dots, \boldsymbol{\psi}_p^{(j)})$
 - \vdots
 - Generate $\boldsymbol{\psi}_p^{(j+1)}$ from $\pi(\boldsymbol{\psi}_p | \boldsymbol{\psi}_1^{(j+1)}, \dots, \boldsymbol{\psi}_{p-1}^{(j+1)})$.
3. Return the values $\{\boldsymbol{\psi}^{(1)}, \boldsymbol{\psi}^{(2)}, \dots, \boldsymbol{\psi}^{(M)}\}$.

It follows that the transition density of moving from $\boldsymbol{\psi}_k^{(j)}$ to $\boldsymbol{\psi}_k^{(j+1)}$ is given by

$$\pi(\boldsymbol{\psi}_k | \boldsymbol{\psi}_1^{(j+1)}, \dots, \boldsymbol{\psi}_{k-1}^{(j+1)}, \boldsymbol{\psi}_{k+1}^{(j)}, \dots, \boldsymbol{\psi}_p^{(j)})$$

since when the k th block is reached, the previous $(k-1)$ blocks have been updated. Thus, the transition density of the chain, under the maintained assumption that π is absolutely continuous, is given by the product of transition kernels for each block:

$$K(\boldsymbol{\psi}, \boldsymbol{\psi}') = \prod_{k=1}^p \pi(\boldsymbol{\psi}_k | \boldsymbol{\psi}_1^{(j+1)}, \dots, \boldsymbol{\psi}_{k-1}^{(j+1)}, \boldsymbol{\psi}_{k+1}^{(j)}, \dots, \boldsymbol{\psi}_p^{(j)}) . \quad (3.23)$$

To illustrate the manner in which the blocks are revised, we consider a two block case, each with a single component, and trace out in Fig. 3.4 a possible

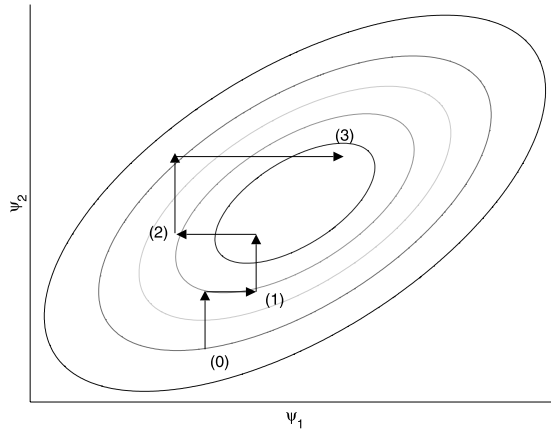


Figure 3.4. Gibbs sampling algorithm in two dimensions starting from an initial point and then completing three iterations

trajectory of the sampling algorithm. The contours in the plot represent the joint distribution of $\boldsymbol{\psi}$ and the labels “(0)”, “(1)” etc., denote the simulated values. Note that one iteration of the algorithm is completed after both components are revised. Also notice that each component is revised along the direction of the coordinate axes. This feature can be a source of problems if the two components are highly correlated because then the contours get compressed and movements along the coordinate axes tend to produce small moves. We return to this issue below.

Invariance of the Gibbs Markov Chain

3.4.2

The Gibbs transition kernel is invariant by construction. This is a consequence of the fact that the Gibbs algorithm is a special case of the multiple-block M–H algorithm which is invariant, as was established in the last section. Invariance can also be confirmed directly. Consider for simplicity a two block sampler with transition kernel density

$$K(\boldsymbol{\psi}, \boldsymbol{\psi}') = \pi(\boldsymbol{\psi}'_1 | \boldsymbol{\psi}_2) \pi(\boldsymbol{\psi}'_2 | \boldsymbol{\psi}'_1).$$

To check invariance we have to show that

$$\begin{aligned} & \int K(\boldsymbol{\psi}, \boldsymbol{\psi}') \pi(\boldsymbol{\psi}_1, \boldsymbol{\psi}_2) d\boldsymbol{\psi}_1 d\boldsymbol{\psi}_2 \\ &= \int \pi(\boldsymbol{\psi}'_1 | \boldsymbol{\psi}_2) \pi(\boldsymbol{\psi}'_2 | \boldsymbol{\psi}'_1) \pi(\boldsymbol{\psi}_1, \boldsymbol{\psi}_2) d\boldsymbol{\psi}_1 d\boldsymbol{\psi}_2 \end{aligned}$$

is equal to $\pi(\boldsymbol{\psi}'_1, \boldsymbol{\psi}'_2)$. This holds because $\pi(\boldsymbol{\psi}'_2 | \boldsymbol{\psi}'_1)$ comes out of the integral, and the integral over $\boldsymbol{\psi}_1$ and $\boldsymbol{\psi}_2$ produces $\pi(\boldsymbol{\psi}'_1)$. This calculation can be extended to any number of blocks. It may be noted that the Gibbs Markov chain is not reversible. Reversible Gibbs samplers are discussed by Liu et al. (1995).

Sufficient Conditions for Convergence

3.4.3

Under rather general conditions, the Markov chain generated by the Gibbs sampling algorithm converges to the target density as the number of iterations become large. Formally, if we let $K(\boldsymbol{\psi}, \boldsymbol{\psi}')$ represent the transition density of the Gibbs algorithm and let $K^{(M)}(\boldsymbol{\psi}_0, \boldsymbol{\psi}')$ be the density of the draw $\boldsymbol{\psi}'$ after M iterations given the starting value $\boldsymbol{\psi}_0$, then

$$\|K^{(M)}(\boldsymbol{\psi}^{(0)}, \boldsymbol{\psi}') - \pi(\boldsymbol{\psi}')\| \rightarrow 0, \quad \text{as } M \rightarrow \infty. \quad (3.24)$$

Roberts and Smith (1994) (see also Chan, 1993) have shown that the conditions of Proposition 2 are satisfied under the following conditions: (1) $\pi(\boldsymbol{\psi}) > 0$ implies there exists an open neighborhood $N_{\boldsymbol{\psi}}$ containing $\boldsymbol{\psi}$ and $\varepsilon > 0$ such that, for all $\boldsymbol{\psi}' \in N_{\boldsymbol{\psi}}$, $\pi(\boldsymbol{\psi}') \geq \varepsilon > 0$; (2) $\int f(\boldsymbol{\psi}) d\boldsymbol{\psi}_k$ is locally bounded for all k , where $\boldsymbol{\psi}_k$ is the k th block of parameters; and (3) the support of $\boldsymbol{\psi}$ is arc connected.

These conditions are satisfied in a wide range of problems.

3.4.4 Example: Simulating a Truncated Multivariate Normal

Consider the question of sampling a trivariate normal distribution truncated to the positive orthant. In particular, suppose that the target distribution is

$$\begin{aligned}\pi(\boldsymbol{\psi}) &= \frac{1}{\Pr(\boldsymbol{\psi} \in A)} f_N(\boldsymbol{\mu}, \boldsymbol{\Sigma}) I(\boldsymbol{\psi} \in A) \\ &\propto f_N(\boldsymbol{\mu}, \boldsymbol{\Sigma}) I(\boldsymbol{\psi} \in A)\end{aligned}$$

where $\boldsymbol{\mu} = (0.5, 1, 1.5)'$, $\boldsymbol{\Sigma}$ is in equi-correlated form with units on the diagonal and 0.7 on the off-diagonal, $A = (0, \infty) \times (0, \infty) \times (0, \infty)$ and $\Pr(\boldsymbol{\psi} \in A)$ is the normalizing constant which is difficult to compute. In this case, the Gibbs sampler is defined with the blocks ψ_1 , ψ_2 , ψ_3 and the full conditional distributions

$$\pi(\psi_1 | \psi_2, \psi_3); \quad \pi(\psi_2 | \psi_1, \psi_3); \quad \pi(\psi_3 | \psi_1, \psi_2),$$

where each of these full conditional distributions is univariate truncated normal restricted to the interval $(0, \infty)$:

$$\pi(\psi_k | \boldsymbol{\psi}_{-k}) \propto f_N(\psi_k | \boldsymbol{\mu}_k + \mathbf{C}_k' \boldsymbol{\Sigma}_{-k}^{-1} (\boldsymbol{\psi}_{-k} - \boldsymbol{\mu}_{-k}), \boldsymbol{\Sigma}_k - \mathbf{C}_k' \boldsymbol{\Sigma}_{-k}^{-1} \mathbf{C}_k) I(\psi_k \in (0, \infty)), \quad (3.25)$$

$\mathbf{C}_k = \text{Cov}(\psi_k, \boldsymbol{\psi}_{-k})$, $\boldsymbol{\Sigma}_{-k} = \text{Var}(\boldsymbol{\psi}_{-k})$ and $\boldsymbol{\mu}_{-k} = E(\boldsymbol{\psi}_{-k})$. Figure 3.5 gives the marginal distribution of each component of $\boldsymbol{\psi}_k$ from a Gibbs sampling run of $M = 10,000$ iterations with a burn-in of 100 cycles. The figures include both the histograms of the sampled values and the Rao-Blackwellized estimates of the marginal densities (see Sect. 3.6 below) based on the averaging of (3.25) over the simulated values of $\boldsymbol{\psi}_{-k}$. The agreement between the two density estimates is close. In the bottom panel of Fig. 3.5 we plot the autocorrelation function of the sampled draws. The rapid decline in the autocorrelations for higher lags indicates that the sampler is mixing well.

3.5 MCMC Sampling with Latent Variables

In designing MCMC simulations, it is sometimes helpful to modify the target distribution by introducing latent variables or auxiliary variables into the sampling. This idea was called data augmentation by Tanner and Wong (1987) in the context of missing data problems. Slice sampling, which we do not discuss in this chapter, is a particular way of introducing auxiliary variables into the sampling, for example see Damien et al. (1999).

To fix notations, suppose that \mathbf{z} denotes a vector of latent variables and let the modified target distribution be $\pi(\boldsymbol{\psi}, \mathbf{z})$. If the latent variables are tactically introduced, the conditional distribution of $\boldsymbol{\psi}$ (or sub components of $\boldsymbol{\psi}$) given \mathbf{z}

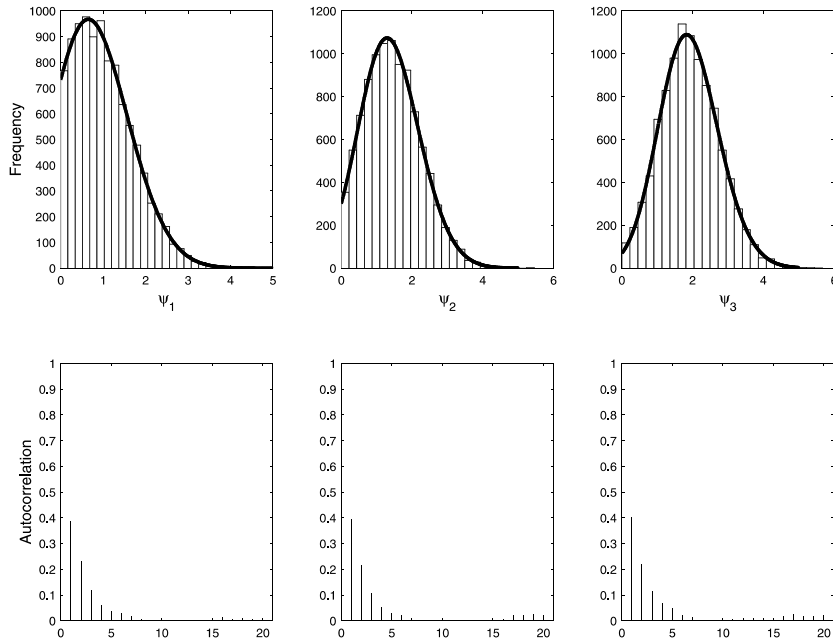


Figure 3.5. Marginal distributions of ψ in truncated multivariate normal example (*top panel*). Histograms of the sampled values and Rao-Blackwellized estimates of the densities are shown. Autocorrelation plots of the Gibbs MCMC chain are in the *bottom panel*. Graphs are based on 10,000 iterations following a burn-in of 500 cycles

may be easy to derive. Then, a multiple-block M-H simulation is conducted with the blocks ψ and z leading to the sample

$$(\psi^{(n_0+1)}, z^{(n_0+1)}), \dots, (\psi^{(n_0+M)}, z^{(n_0+M)}) \sim \pi(\psi, z),$$

where the draws on ψ , ignoring those on the latent data, are from $\pi(\psi)$, as required.

To demonstrate this technique in action, we return to the probit regression example discussed in Sect. 3.3.2 to show how a MCMC sampler can be developed with the help of latent variables. The approach, introduced by Albert and Chib (1993), capitalizes on the simplifications afforded by introducing latent or auxiliary data into the sampling.

The model is rewritten as

$$\begin{aligned} z_i | \beta &\sim N(x_i' \beta, 1), \\ y_i &= I[z_i > 0], \quad i \leq n, \\ \beta &\sim N_k(\beta_0, B_0). \end{aligned} \tag{3.26}$$

This specification is equivalent to the probit regression model since

$$\Pr(y_i = 1 | x_i, \beta) = \Pr(z_i > 0 | x_i, \beta) = \Phi(x_i' \beta).$$

Now the Albert–Chib algorithm proceeds with the sampling of the full conditional distributions

$$\boldsymbol{\beta} | \mathbf{y}, \{z_i\}; \quad \{z_i\} | \mathbf{y}, \boldsymbol{\beta},$$

where both these distributions are tractable (i.e., requiring no M–H steps). In particular, the distribution of $\boldsymbol{\beta}$ conditioned on the latent data becomes independent of the observed data and has the same form as in the Gaussian linear regression model with the response data given by $\{z_i\}$ and is multivariate normal with mean $\hat{\boldsymbol{\beta}} = \mathbf{B}(\mathbf{B}_0^{-1} \boldsymbol{\beta}_0 + \sum_{i=1}^n \mathbf{x}_i z_i)$ and variance matrix $\mathbf{B} = (\mathbf{B}_0^{-1} + \sum_{i=1}^n \mathbf{x}_i \mathbf{x}_i')^{-1}$. Next, the distribution of the latent data conditioned on the data and the parameters factor into a set of n independent distributions with each depending on the data through y_i :

$$\{z_i\} | \mathbf{y}, \boldsymbol{\beta} \stackrel{d}{=} \prod_{i=1}^n z_i | y_i, \boldsymbol{\beta},$$

where the distribution $z_i | y_i, \boldsymbol{\beta}$ is the normal distribution $z_i | \boldsymbol{\beta}$ truncated by the knowledge of y_i ; if $y_i = 0$, then $z_i \leq 0$ and if $y_i = 1$, then $z_i > 0$. Thus, one samples z_i from $\mathcal{TN}_{(-\infty, 0]}(\mathbf{x}_i' \boldsymbol{\beta}, 1)$ if $y_i = 0$ and from $\mathcal{TN}_{(0, \infty)}(\mathbf{x}_i' \boldsymbol{\beta}, 1)$ if $y_i = 1$, where $\mathcal{TN}_{(a, b)}(\boldsymbol{\mu}, \boldsymbol{\sigma}^2)$ denotes the $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\sigma}^2)$ distribution truncated to the region (a, b) .

The results, based on 5000 MCMC draws beyond a burn-in of a 100 iterations, are reported in Fig. 3.4. The results are close to those presented above, especially to the ones from the tailored M–H chain.

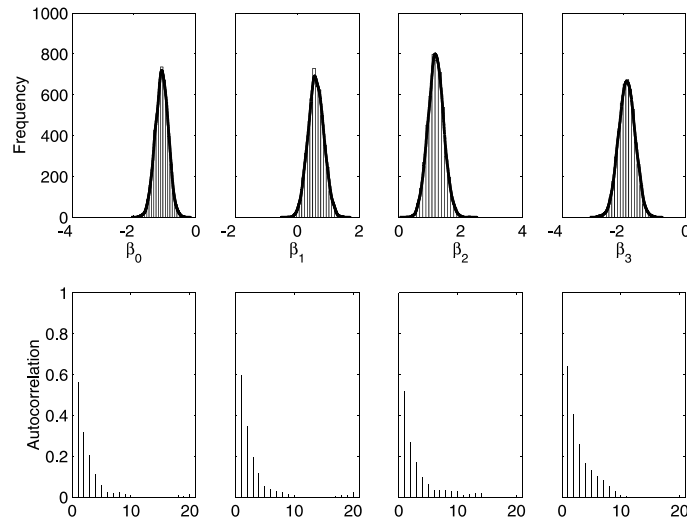


Figure 3.6. Caesarean data with Albert–Chib algorithm: Marginal posterior densities (*top panel*) and autocorrelation plot (*bottom panel*)

Estimation of Density Ordinates

We mention that if the full conditional densities are available, whether in the context of the multiple-block M–H algorithm or that of the Gibbs sampler, then the MCMC output can be used to estimate posterior marginal density functions (Tanner and Wong, 1987; Gelfand and Smith, 1990). We exploit the fact that the marginal density of $\boldsymbol{\psi}_k$ at the point $\boldsymbol{\psi}_k^*$ is

$$\pi(\boldsymbol{\psi}_k^*) = \int \pi(\boldsymbol{\psi}_k^* | \boldsymbol{\psi}_{-k}) \pi(\boldsymbol{\psi}_{-k}) d\boldsymbol{\psi}_{-k},$$

where as before $\boldsymbol{\psi}_{-k} = \boldsymbol{\psi} \setminus \boldsymbol{\psi}_k$. Provided the normalizing constant of $\pi(\boldsymbol{\psi}_k^* | \boldsymbol{\psi}_{-k})$ is known, an estimate of the marginal density is available as an average of the full conditional density over the simulated values of $\boldsymbol{\psi}_{-k}$:

$$\hat{\pi}(\boldsymbol{\psi}_k^*) = M^{-1} \sum_{j=1}^M \pi(\boldsymbol{\psi}_k^* | \boldsymbol{\psi}_{-k}^{(j)}).$$

Under the assumptions of Proposition 1,

$$M^{-1} \sum_{j=1}^M \pi(\boldsymbol{\psi}_k^* | \boldsymbol{\psi}_{-k}^{(j)}) \rightarrow \pi(\boldsymbol{\psi}_k^*), \quad \text{as } M \rightarrow \infty.$$

Gelfand and Smith (1990) refer to this approach as Rao–Blackwellization because of the connections with the Rao–Blackwell theorem in classical statistics. That connection is more clearly seen in the context of estimating (say) the mean of $\boldsymbol{\psi}_k$, $E(\boldsymbol{\psi}_k) = \int \boldsymbol{\psi}_k \pi(\boldsymbol{\psi}_k) d\boldsymbol{\psi}_k$. By the law of the iterated expectation,

$$E(\boldsymbol{\psi}_k) = E\{E(\boldsymbol{\psi}_k | \boldsymbol{\psi}_{-k})\}$$

and therefore the estimates

$$M^{-1} \sum_{j=1}^M \boldsymbol{\psi}_k^j$$

and

$$M^{-1} \sum_{j=1}^M E(\boldsymbol{\psi}_k | \boldsymbol{\psi}_{-k}^{(j)})$$

both converge to $E(\boldsymbol{\psi}_k)$ as $M \rightarrow \infty$. Under *iid* sampling, and under Markov sampling provided some conditions are satisfied – see Liu et al. (1994), Casella and Robert (1996) and Robert and Casella (1999), it can be shown that the variance of the latter estimate is smaller than that of the former. Thus, it can help to average the conditional mean $E(\boldsymbol{\psi}_k | \boldsymbol{\psi}_{-k})$, if that were available, rather than average the draws directly. Gelfand and Smith (1990) appeal to this analogy to argue that the

Rao–Blackwellized estimate of the density is preferable to that based on the method of kernel smoothing. Chib (1995) extends the Rao–Blackwellization approach to estimate reduced conditional ordinates defined as the density of $\boldsymbol{\psi}_k$ conditioned on one or more of the remaining blocks. Finally, Chen (1994) provides an importance weighted estimate of the marginal density for cases where the conditional posterior density does not have a known normalizing constant. Chen’s estimator is based on the identity

$$\pi(\boldsymbol{\psi}_k^*) = \int w(\boldsymbol{\psi}_k | \boldsymbol{\psi}_{-k}) \frac{\pi(\boldsymbol{\psi}_k^*, \boldsymbol{\psi}_{-k})}{\pi(\boldsymbol{\psi}_k, \boldsymbol{\psi}_{-k})} \pi(\boldsymbol{\psi}) d\boldsymbol{\psi},$$

where $w(\boldsymbol{\psi}_k | \boldsymbol{\psi}_{-k})$ is a completely known conditional density whose support is equal to the support of the full conditional density $\pi(\boldsymbol{\psi}_k | \boldsymbol{\psi}_{-k})$. In this form, the normalizing constant of the full conditional density is not required and given a sample of draws $\{\boldsymbol{\psi}^{(1)}, \dots, \boldsymbol{\psi}^{(M)}\}$ from $\pi(\boldsymbol{\psi})$, a Monte Carlo estimate of the marginal density is given by

$$\hat{\pi}(\boldsymbol{\psi}_k^*) = M^{-1} \sum_{j=1}^M w(\boldsymbol{\psi}_k^{(j)} | \boldsymbol{\psi}_{-k}^{(j)}) \frac{\pi(\boldsymbol{\psi}_k^*, \boldsymbol{\psi}_{-k}^{(j)})}{\pi(\boldsymbol{\psi}_k^{(j)}, \boldsymbol{\psi}_{-k}^{(j)})}.$$

Chen (1994) discusses the choice of the conditional density w . Since it depends on $\boldsymbol{\psi}_{-k}$, the choice of w will vary from one sampled draw to the next.

3.7

Sampler Performance and Diagnostics

In implementing a MCMC method it is important to assess the performance of the sampling algorithm to determine the rate of mixing and the size of the burn-in, both having implications for the number of iterations required to get reliable answers. A large literature has emerged on these issues, for example, Robert (1995), Tanner (1996, Sect. 6.3), Cowles and Carlin (1996), Gammernann (1997, Sect. 5.4) and Robert and Casella (1999), but the ideas, although related in many ways, have not coalesced into a single prescription.

One approach for determining sampler performance and the size of the burn-in time is to employ analytical methods to the specified Markov chain, prior to sampling. This approach is exemplified in the work of, for example, Polson (1996), Roberts and Tweedie (1996) and Rosenthal (1995). Two factors have inhibited the growth and application of these methods. The first is that the calculations are difficult and problem-specific and, second, the upper bounds for the burn-in that emerge from such calculations are usually conservative.

At this time the more popular approach is to utilize the sampled draws to assess both the performance of the algorithm and its approach to the invariant distribution. Several such relatively informal methods are available. Gelfand and Smith (1990) recommend monitoring the evolution of the quantiles as the sampling

proceeds. Another useful diagnostic, one that is perhaps the most direct, are autocorrelation plots (and autocorrelation times) of the sampled output. Slowly decaying correlations indicate problems with the mixing of the chain. It is also useful in connection with M–H Markov chains to monitor the acceptance rate of the proposal values with low rates implying “stickiness” in the sampled values and thus a slower approach to the invariant distribution.

Somewhat more formal sample-based diagnostics are summarized in the CODA routines provided by Best et al. (1995). Although these diagnostics often go under the name “convergence diagnostics” they are in principle approaches that detect lack of convergence. Detection of convergence based entirely on the sampled output, without analysis of the target distribution, is perhaps impossible. Cowles and Carlin (1996) discuss and evaluate thirteen such diagnostics (for example, those proposed by Geweke, 1992, Raftery and Lewis, 1992, Ritter and Tanner, 1992, Gelman and Rubin, 1992, Gelman and Rubin, 1992, and Zellner and Min, 1995, amongst others) without arriving at a consensus. Difficulties in evaluating these methods stem from the fact that some of these methods apply only to Gibbs Markov chains (for example, those of Ritter and Tanner, 1992, and Zellner and Min, 1995) while others are based on the output not just of a single chain but on that of multiple chains specifically run from “disparate starting values” as in the method of Gelman and Rubin (1992). Finally, some methods assess the behavior of univariate moment estimates (as in the approach of Geweke, 1992, and Gelman and Rubin, 1992) while others are concerned with the behavior of the entire transition kernel (as in Ritter and Tanner, 1992, and Zellner and Min, 1995).

Strategies for Improving Mixing

3.8

In practice, while implementing MCMC methods it is important to construct samplers that mix well, where mixing is measured by the autocorrelation time, because such samplers can be expected to converge more quickly to the invariant distribution. Over the years a number of different recipes for designing samplers with low autocorrelation times have been proposed although it may sometimes be difficult, because of the complexity of the problem, to apply any of these recipes.

Choice of Blocking

3.8.1

As a general rule, sets of parameters that are highly correlated should be treated as one block when applying the multiple-block M–H algorithm. Otherwise, it would be difficult to develop proposal densities that lead to large moves through the support of the target distribution.

Blocks can be combined by the method of composition. For example, suppose that $\boldsymbol{\psi}_1$, $\boldsymbol{\psi}_2$ and $\boldsymbol{\psi}_3$ denote three blocks and that the distribution $\boldsymbol{\psi}_1|\boldsymbol{\psi}_3$ is tractable (i.e., can be sampled directly). Then, the blocks $(\boldsymbol{\psi}_1, \boldsymbol{\psi}_2)$ can be collapsed by first sampling $\boldsymbol{\psi}_1$ from $\boldsymbol{\psi}_1|\boldsymbol{\psi}_3$ followed by $\boldsymbol{\psi}_2$ from $\boldsymbol{\psi}_2|\boldsymbol{\psi}_1, \boldsymbol{\psi}_3$. This amounts to

a two block MCMC algorithm. In addition, if it is possible to sample $(\boldsymbol{\psi}_1, \boldsymbol{\psi}_2)$ marginalized over $\boldsymbol{\psi}_3$ then the number of blocks is reduced to one. Liu et al. (1994) discuss the value of these strategies in the context of a three-block Gibbs MCMC chains. Roberts and Sahu (1997) provide further discussion of the role of blocking in the context of Gibbs Markov chains used to sample multivariate normal target distributions.

3.8.2 Tuning the Proposal Density

As mentioned above, the proposal density in a M–H algorithm has an important bearing on the mixing of the MCMC chain. Fortunately, one has great flexibility in the choice of candidate generating density and it is possible to adapt the choice to the given problem. For example, Chib et al. (1998) develop and compare four different choices in longitudinal random effects models for count data. In this problem, each cluster (or individual) has its own random effects and each of these has to be sampled from an intractable target distribution. If one lets n denote the number of clusters, where n is typically large, say in excess of a thousand, then the number of blocks in the MCMC implementation is $n + 3$ (n for each of the random effect distributions, two for the fixed effects and one for the variance components matrix). For this problem, the multiple-block M–H algorithm requires $n + 1$ M–H steps within one iteration of the algorithm. Tailored proposal densities are therefore computationally expensive but one can use a mixture of proposal densities where a less demanding proposal, for example a random walk proposal, is combined with the tailored proposal to sample each of the n random effect target distributions. Further discussion of mixture proposal densities is contained in Tierney (1994).

3.8.3 Other Strategies

Other approaches have also been discussed in the literature. Marinari and Parisi (1992) develop the simulated tempering method whereas Geyer and Thompson (1995) develop a related technique that they call the Metropolis-coupled MCMC method. Both these approaches rely on a series of transition kernels $\{K_1, \dots, K_m\}$ where only K_1 has π^* as the stationary distribution. The other kernels have equilibrium distributions π_i , which Geyer and Thompson (1995) take to be $\pi_i(\boldsymbol{\psi}) = \pi(\boldsymbol{\psi})^{1/i}$, $i = 2, \dots, m$. This specification produces a set of target distributions that have higher variance than π^* . Once the transition kernels and equilibrium distributions are specified then the Metropolis-coupled MCMC method requires that each of the m kernels be used in parallel. At each iteration, after the m draws have been obtained, one randomly selects two chains to see if the states should be swapped. The probability of swap is based on the M–H acceptance condition. At the conclusion of the sampling, inference is based on the sequence of draws that correspond to the distribution π^* . These methods promote rapid mixing because draws from the various “flatter” target densities have a chance of being swapped with the draws from the base kernel K_1 . Thus, variates that are

unlikely under the transition K_1 have a chance of being included in the chain, leading to more rapid exploration of the parameter space.

Concluding Remarks

3.9

In this survey we have provided an outline of Markov chain Monte Carlo methods. These methods provide a set of general recipes for sampling intractable multivariate distributions and have proved vital in the recent virtually revolutionary evolution and growth of Bayesian statistics. Refinements and extensions of these methods continue to occur. Two recent developments are the slice sampling method discussed by Mira and Tierney (2002), Damien et al. (1999) and Roberts and Rosenthal (1999) and the perfect sampling method proposed by Propp and Wilson (1996). The slice sampling method is based on the introduction of auxiliary uniform random variables to simplify the sampling and improve mixing while the perfect sampling method uses Markov chain coupling to generate an exact draw from the target distribution.

References

- Albert, J. and Chib, S. (1993). Bayesian analysis of binary and polychotomous response data. *Journal of the American Statistical Association*, 88: 669–679.
- Besag, J. (1974). Spatial interaction and the statistical analysis of lattice systems (with discussion). *Journal of the Royal Statistical Society B*, 36: 192–236.
- Besag, J., Green, E., Higdon, D. and Mengersen, K.L. (1995). Bayesian Computation and Stochastic Systems (with discussion). *Statistical Science*, 10: 3–66.
- Best, N.G., Cowles, M.K. and Vines, S.K. (1995). CODA: Convergence diagnostics and output analysis software for Gibbs sampling. Technical report, Cambridge MRC Biostatistics Unit.
- Carlin, B.P. and Louis, T. (2000). *Bayes and Empirical Bayes Methods for Data Analysis, 2nd ed*, Chapman and Hall, London.
- Casella, G. and Robert, C.P. (1996). Rao–Blackwellization of sampling schemes. *Biometrika*, 83: 81–94.
- Chan, K.S. (1993). Asymptotic behavior of the Gibbs sampler. *Journal of the American Statistical Association*, 88: 320–326.
- Chan, K.S. and Ledolter, J. (1995). Monte Carlo EM estimation for time series models involving counts. *Journal of the American Statistical Association*, 90: 242–252.
- Chen, M-H. (1994). Importance-weighted marginal Bayesian posterior density estimation. *Journal of the American Statistical Association*, 89: 818–824.
- Chen, M-H., Shao, Qi-M. and Ibrahim, J.G. (2000), *Monte Carlo Methods in Bayesian Computation*, Springer Verlag, New York.
- Chib, S. (1995). Marginal likelihood from the Gibbs output. *Journal of the American Statistical Association*, 90: 1313–1321.

- Chib, S. (2001). Markov Chain Monte Carlo Methods: Computation and Inference. In Heckman, J.J. and Leamer, E. (eds), *Handbook of Econometrics, Volume 5*, pp. 3569–3649, North Holland, Amsterdam.
- Chib, S. and Greenberg, E. (1994). Bayes inference for regression models with ARMA(p, q) errors. *Journal of Econometrics*, 64: 183–206.
- Chib, S. and Greenberg, E. (1995). Understanding the Metropolis–Hastings algorithm. *American Statistician*, 49: 327–335.
- Chib, S. and Greenberg, E. (1996). Markov chain Monte Carlo simulation methods in econometrics. *Econometric Theory*, 12: 409–431.
- Chib, S. Greenberg, E. and Winkleman, R. (1998). Posterior simulation and Bayes factors in panel count data models. *Journal of Econometrics*, 86, 33–54.
- Chib, S. and Jeliazkov, I. (2001). Marginal likelihood from the Metropolis–Hastings output. *Journal of the American Statistical Association*, 96: 270–281.
- Congdon, P. (2001). *Bayesian Statistical Modeling*, John Wiley, Chichester.
- Cowles, M.K. and Carlin, B. (1996). Markov chain Monte Carlo convergence diagnostics: A comparative review. *Journal of the American Statistical Association*, 91: 883–904.
- Damien, P., Wakefield, J. and Walker, S. (1999). Gibbs Sampling for Bayesian non-conjugate and hierarchical models using auxiliary variables. *Journal of the Royal Statistical Society B*, 61, 331–344.
- Fahrmeir, L. and Tutz, G. (1997). *Multivariate Statistical Modeling Based on Generalized Linear Models*. Springer Verlag, New York.
- Gamerman, D. (1997). *Markov chain Monte Carlo: Stochastic Simulation for Bayesian Inference*. Chapman and Hall, London.
- Gelfand, A.E. and Smith, A.F.M. (1990). Sampling-based approaches to calculating marginal densities. *Journal of the American Statistical Association*, 85: 398–409.
- Gelman, A. and Rubin, D.B. (1992). Inference from iterative simulation using multiple sequences. *Statistical Science*, 4: 457–472.
- Gelman, A., Meng, X.L., Stern, H.S. and Rubin, D.B. (2003). *Bayesian Data Analysis*, (2nd ed), Chapman and Hall, London.
- Geman, S. and Geman, D. (1984). Stochastic relaxation, Gibbs distributions and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12: 609–628.
- Geweke, J. (1989). Bayesian inference in econometric models using Monte Carlo integration. *Econometrica*, 57: 1317–1340.
- Geweke, J. (1992). Evaluating the accuracy of sampling-based approaches to the calculation of posterior moments. In Bernardo, J.M., Berger, J.O., Dawid, A.P. and Smith, A.F.M. (eds), *Bayesian Statistics*, pp. 169–193, Oxford University Press, New York.
- Geyer, C. (1992). Practical Markov chain Monte Carlo. *Statistical Science*, 4: 473–482.
- Geyer, C. and Thompson, E.A. (1995). Annealing Markov chain Monte Carlo with Applications to Ancestral Inference. *Journal of American Statistical Association*, 90: 909–920.

- Gilks, W.R., Richardson, S. and Spiegelhalter, D.J. (1996). *Markov Chain Monte Carlo in Practice*, Chapman and Hall, London.
- Hastings, W. K. (1970). Monte Carlo sampling methods using Markov chains and their applications. *Biometrika* 57: 97–109.
- Liu, J.S. (1994). The collapsed Gibbs sampler in Bayesian computations with applications to a gene regulation problem. *Journal of the American Statistical Association*, 89: 958–966.
- Liu, J. S. (2001), *Monte Carlo Strategies in Scientific Computing*, Springer Verlag, New York.
- Liu, J.S., Wong, W.H. and Kong, A. (1994). Covariance structure of the Gibbs Sampler with applications to the comparisons of estimators and data augmentation schemes. *Biometrika*, 81: 27–40.
- Liu, J.S., Wong, W.H. and Kong, A. (1995). Covariance structure and convergence rate of the Gibbs sampler with various scans. *Journal of the Royal Statistical Society B*, 57: 157–169.
- Marinari, E. and Parisi, G. (1992). Simulated tempering: A new Monte Carlo scheme. *Europhysics Letters*, 19: 451–458.
- Mengersen, K.L. and Tweedie, R.L. (1996). Rates of convergence of the Hastings and Metropolis algorithms. *Annals of Statistics*, 24: 101–121.
- Metropolis, N., Rosenbluth, A.W., Rosenbluth, M.N., Teller, A.H. and Teller, E. (1953). Equations of state calculations by fast computing machines. *Journal of Chemical Physics*, 21: 1087–1092.
- Meyn, S.P. and Tweedie, R.L. (1993). *Markov chains and stochastic stability*. Springer-Verlag, London.
- Mira, A. and Tierney, L. (2002). Efficiency and convergence properties of slice samplers. *Scandinavian Journal Of Statistics*, 29: 1–12.
- Nummelin, E. (1984). *General irreducible Markov chains and non-negative operators*. Cambridge, Cambridge University Press.
- Polson, N. G. (1996). Convergence of Markov Chain Monte Carlo algorithms. In Bernardo, J.M., Berger, J.O., Dawid, A.P. and Smith, A.F.M. (eds), *Proceedings of the Fifth Valencia International Conference on Bayesian Statistics*, pp. 297–323, Oxford University Press, Oxford.
- Propp, J.G. and Wilson, D.B. (1996). Exact sampling with coupled Markov chains and applications to statistical mechanics. *Random Structures and Algorithms*, 9: 223–252.
- Raftery, A. E. and Lewis, S.M. (1992). How many iterations in the Gibbs sampler? In Bernardo, J.M., Berger, J.O., Dawid, A.P. and Smith, A.F.M. (eds), *Proceedings of the Fourth Valencia International Conference on Bayesian Statistics*, pp. 763–774, Oxford University Press, New York.
- Ripley, B. (1987). *Stochastic simulation*, John Wiley & Sons, New York.
- Ritter, C. and Tanner, M.A. (1992). Facilitating the Gibbs Sampler: the Gibbs Stopper and the Griddy-Gibbs Sampler. *Journal of the American Statistical Association*, 87: 861–868.
- Robert C.P. (1995). Convergence control methods for Markov chain Monte Carlo algorithms. *Statistical Science*, 10: 231–253.

- Robert, C.P. (2001). *Bayesian Choice*, 2nd ed, Springer Verlag, New York.
- Robert, C.P. and Casella, G. (1999). *Monte Carlo Statistical Methods*, Springer Verlag, New York.
- Roberts, G.O. and Rosenthal, J.S. (1999). Convergence of slice sampler Markov chains. *Journal of the Royal Statistical Society B*, 61: 643–660.
- Roberts, G.O. and Sahu, S.K. (1997). Updating schemes, correlation structure, blocking, and parametrization for the Gibbs sampler. *Journal of the Royal Statistical Society B*, 59: 291–317.
- Roberts, G.O. and Smith, A.F.M. (1994). Some simple conditions for the convergence of the Gibbs sampler and Metropolis–Hastings algorithms. *Stochastic Processes and its Applications*, 49: 207–216.
- Roberts, G.O. and Tweedie, R.L. (1996). Geometric convergence and central limit theorems for multidimensional Hastings and Metropolis algorithms. *Biometrika*, 83: 95–110.
- Rosenthal, J.S. (1995). Minorization conditions and convergence rates for Markov chain Monte Carlo. *Journal of the American Statistical Association*, 90: 558–566.
- Smith, A.F.M. and Roberts, G.O. (1993). Bayesian computation via the Gibbs sampler and related Markov chain Monte Carlo methods. *Journal of the Royal Statistical Society B*, 55: 3–24.
- Tanner, M.A. (1996). *Tools for Statistical Inference*, 3rd ed, Springer-Verlag, New York.
- Tanner, M.A. and Wong, W.H. (1987). The calculation of posterior distributions by data augmentation. *Journal of the American Statistical Association*, 82: 528–549.
- Tierney, L. (1994). Markov chains for exploring posterior distributions (with discussion). *Annals of Statistics*, 22: 1701–1762.
- Zellner, A. and Min, C. (1995). Gibbs sampler convergence criteria. *Journal of the American Statistical Association*, 90: 921–927.

Numerical Linear Algebra

II.4

Lenka Čížková, Pavel Čížek

4.1	<i>Matrix Decompositions</i>	104
	Cholesky Decomposition	105
	LU Decomposition	106
	QR Decomposition	108
	Singular Value Decomposition	114
	Matrix Inversion	115
4.2	<i>Direct Methods for Solving Linear Systems</i>	116
	Gauss–Jordan Elimination	117
	Iterative Refinement	119
4.3	<i>Iterative Methods for Solving Linear Systems</i>	120
	General Principle of Iterative Methods for Linear Systems	120
	Jacobi Method	122
	Gauss–Seidel Method	122
	Successive Overrelaxation Method	123
	Gradient Methods	124
4.4	<i>Eigenvalues and Eigenvectors</i>	126
	Power Method	127
	Jacobi Method	127
	Givens and Householder Reductions	128
	QR Method	128
	LR Method	129
	Inverse Iterations	129
4.5	<i>Sparse Matrices</i>	129
	Storage Schemes for Sparse Matrices	130
	Methods for Sparse Matrices	131

Many methods of computational statistics lead to matrix-algebra or numerical-mathematics problems. For example, the least squares method in linear regression reduces to solving a system of linear equations, see Chap. III.8. The principal components method is based on finding eigenvalues and eigenvectors of a matrix, see Chap. III.6. Nonlinear optimization methods such as Newton's method often employ the inversion of a Hessian matrix. In all these cases, we need numerical linear algebra.

Usually, one has a data matrix X of (explanatory) variables, and in the case of regression, a data vector y for dependent variable. Then the matrix defining a system of equations, being inverted or decomposed typically corresponds to X or $X^T X$. We refer to the matrix being analyzed as $A = \{A_{ij}\}_{i=1, j=1}^{m, n} \in \mathbb{R}^{m \times n}$ and to its columns as $A_k = \{A_{ik}\}_{i=1}^m, k = 1, \dots, n$. In the case of linear equations, $b = \{b_i\}_{i=1}^n \in \mathbb{R}^n$ represents the right-hand side throughout this chapter. Further, the eigenvalues and singular values of A are denoted by λ_i and σ_i , respectively, and the corresponding eigenvectors $g_i, i = 1, \dots, n$. Finally, we denote the $n \times n$ identity and zero matrices by I_n and 0_n , respectively.

In this chapter, we first study various matrix decompositions (Sect. 4.1), which facilitate numerically stable algorithms for solving systems of linear equations and matrix inversions. Next, we discuss specific direct and iterative methods for solving linear systems (Sects. 4.2 and 4.3). Further, we concentrate on finding eigenvalues and eigenvectors of a matrix in Sect. 4.4. Finally, we provide an overview of numerical methods for large problems with sparse matrices (Sect. 4.5).

Let us note that most of the mentioned methods work under specific conditions given in existence theorems, which we state without proofs. Unless said otherwise, the proofs can be found in Harville (1997), for instance. Moreover, implementations of the algorithms described here can be found, for example, in Anderson et al. (1999) and Press et al. (1992).

4.1

Matrix Decompositions

This section covers relevant matrix decompositions and basic numerical methods. Decompositions provide a numerically stable way to solve a system of linear equations, as shown already in Wampler (1970), and to invert a matrix. Additionally, they provide an important tool for analyzing the numerical stability of a system.

Some of most frequently used decompositions are the Cholesky, QR, LU, and SVD decompositions. We start with the Cholesky and LU decompositions, which work only with positive definite and nonsingular diagonally dominant square matrices, respectively (Sects. 4.1.1 and 4.1.2). Later, we explore more general and in statistics more widely used QR and SVD decompositions, which can be applied to any matrix (Sects. 4.1.3 and 4.1.4). Finally, we briefly describe the use of decompositions for matrix inversion, although one rarely needs to invert a matrix (Sect. 4.1.5). Monographs Gentle (1998), Harville (1997), Higham (2002) and Stewart (1998) include extensive discussions of matrix decompositions.

All mentioned decompositions allow us to transform a general system of linear equations to a system with an upper triangular, a diagonal, or a lower triangular coefficient matrix: $Ux = b$, $Dx = b$, or $Lx = b$, respectively. Such systems are easy to solve with a very high accuracy by back substitution, see Higham (1989). Assuming the respective coefficient matrix A has a full rank, one can find a solution of $Ux = b$, where $U = \{U_{ij}\}_{i=1,j=1}^n$, by evaluating

$$x_i = U_{ii}^{-1} \left(b_i - \sum_{j=i+1}^n U_{ij}x_j \right) \tag{4.1}$$

for $i = n, \dots, 1$. Analogously for $Lx = b$, where $L = \{L_{ij}\}_{i=1,j=1}^n$, one evaluates for $i = 1, \dots, n$

$$x_i = L_{ii}^{-1} \left(b_i - \sum_{j=1}^{i-1} L_{ij}x_j \right). \tag{4.2}$$

For discussion of systems of equations that do not have a full rank, see for example Higham (2002).

Cholesky Decomposition

4.1.1

The Cholesky factorization, first published by Benoit (1924), was originally developed to solve least squares problems in geodesy and topography. This factorization, in statistics also referred to as “square root method,” is a triangular decomposition. Providing matrix A is positive definite, the Cholesky decomposition finds a triangular matrix U that multiplied by its own transpose leads back to matrix A . That is, U can be thought of as a square root of A .

Theorem 1 Let matrix $A \in \mathbb{R}^{n \times n}$ be symmetric and positive definite. Then there exists a unique upper triangular matrix U with positive diagonal elements such that $A = U^T U$.

1

The matrix U is called the Cholesky factor of A and the relation $A = U^T U$ is called the Cholesky factorization.

Obviously, decomposing a system $Ax = b$ to $U^T Ux = b$ allows us to solve two triangular systems: $U^T z = b$ for z and then $Ux = z$ for x . This is similar to the original Gauss approach for solving a positive definite system of normal equations $X^T Xx = X^T b$. Gauss solved the normal equations by a symmetry-preserving elimination and used the back substitution to solve for x .

Let us now describe the algorithm for finding the Cholesky decomposition, which is illustrated on Fig. 4.1. One of the interesting features of the algorithm is that in the i th iteration we obtain the Cholesky decomposition of the i th leading principal minor of A , $\{A_{kl}\}_{k=1,l=1}^i$.

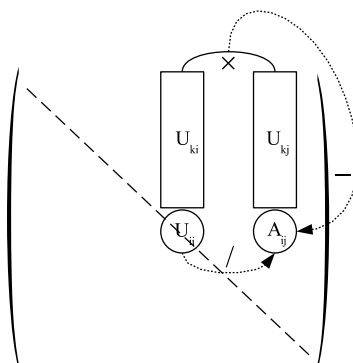


Figure 4.1. Rowwise Cholesky algorithm

Algorithm 1

```

for i=1 to n
   $U_{ii} = (A_{ii} - \sum_{k=1}^{i-1} U_{ki}^2)^{1/2}$ 
  for j=i+1 to n
     $U_{ij} = (A_{ij} - \sum_{k=1}^{i-1} U_{ki}U_{kj}) / U_{ii}$ 
  end
end

```

The Cholesky decomposition described in Algorithm 1 is a numerically stable procedure, see Martin et al. (1965) and Meinguet (1983), which can be at the same time implemented in a very efficient way. Computed values U_{ij} can be stored in place of original A_{ij} , and thus, no extra memory is needed. Moreover, let us note that while Algorithm 1 describes the rowwise decomposition (U is computed row by row), there are also a columnwise version and a version with diagonal pivoting, which is also applicable to semidefinite matrices. Finally, there are also modifications of the algorithm, such as blockwise decomposition, that are suitable for very large problems and parallelization; see Björck (1996), Gallivan et al. (1990) and Nool (1995).

4.1.2 LU Decomposition

The LU decomposition is another method reducing a square matrix A to a product of two triangular matrices (lower triangular L and upper triangular U). Contrary to the Cholesky decomposition, it does not require a positive definite matrix A , but there is no guarantee that $L = U^T$.

Theorem 2 Let the matrix $A \in \mathbb{R}^{n \times n}$ satisfy following conditions:

2

$$A_{11} \neq 0, \quad \det \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \neq 0, \quad \det \begin{pmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{pmatrix} \neq 0, \quad \dots, \quad \det A \neq 0.$$

Then there exists a unique lower triangular matrix L with ones on a diagonal, a unique upper triangular matrix U with ones on a diagonal and a unique diagonal matrix D such that $A = LDU$.

Note that for any nonsingular matrix A there is always a row permutation P such that the permuted matrix PA satisfies the assumptions of Theorem 2. Further, a more frequently used version of this theorem factorizes A to a lower triangular matrix $L' = LD$ and an upper triangular matrix $U' = U$. Finally, Zou (1991) gave alternative conditions for the existence of the LU decomposition: $A \in \mathbb{R}^{n \times n}$ is nonsingular and A^T is diagonally dominant (i.e., $|A_{ii}| \geq \sum_{i=1, i \neq j}^n |A_{ij}|$ for $j = 1, \dots, n$).

Similarly to the Cholesky decomposition, the LU decomposition reduces solving a system of linear equations $Ax = LUx = b$ to solving two triangular systems: $Lz = b$, where $z = Ux$, and $Ux = z$.

Finding the LU decomposition of A is described in Algorithm 2. Since it is equivalent to solving a system of linear equations by the Gauss elimination, which searches just for U and ignores L , we refer a reader to Sect. 4.2, where its advantages (e.g., easy implementation, speed) and disadvantages (e.g., numerical instability without pivoting) are discussed.

Algorithm 2

```

L = 0_n, U = I_n
for i = 1 to n
  for j = i to n
    Lji = Aji - ∑k=1i-1 LjkUki
  end
  for j = i + 1 to n
    Uij = (Aij - ∑k=1i-1 LikUkj) / Lii
  end
end
end
    
```

Finally, note that there are also generalizations of LU to non-square and singular matrices, such as rank revealing LU factorization; see Pan (2000) and Miranian and Gu (2003).

4.1.3 QR Decomposition

One of the most important matrix transformations is the QR decomposition. It splits a general matrix A to an orthonormal matrix Q , that is, a matrix with columns orthogonal to each other and its Euclidian norm equal to 1, and to an upper triangular matrix R . Thus, a suitably chosen orthogonal matrix Q will triangularize the given matrix A .

3 **Theorem 3** Let matrix $A \in \mathbb{R}^{m \times n}$ with $m \geq n$. Then there exist an orthonormal matrix $Q \in \mathbb{R}^{m \times m}$ and an upper triangular matrix $R \in \mathbb{R}^{n \times n}$ with nonnegative diagonal elements such that

$$A = Q \begin{pmatrix} R \\ 0 \end{pmatrix}$$

(the QR decomposition of the matrix A).

If A is a nonsingular square matrix, an even slightly stronger result can be obtained: uniqueness of the QR decomposition.

4 **Theorem 4** Let matrix $A \in \mathbb{R}^{n \times n}$ be nonsingular. Then there exist a unique orthonormal matrix $Q \in \mathbb{R}^{n \times n}$ and a unique upper triangular matrix $R \in \mathbb{R}^{n \times n}$ with positive diagonal elements such that $A = QR$.

The use of the QR decomposition for solving a system of equations $Ax = QRx = b$ consists in multiplying the whole system by the orthonormal matrix Q^T , $Q^T Q = I$, and then solving the remaining upper triangular system $Rx = Q^T b$. This method guarantees numerical stability by minimizing errors caused by machine roundoffs (see the end of this section for details).

The QR decomposition is usually constructed by finding one orthonormal vector (one column of Q) after another. This can be achieved using the so-called elementary orthogonal transformations such as Householder reflections, Householder (1958), or Givens rotations, Givens (1958), that are described in the following subsections. These transformations are related to the solution of the following standard task.

1 **Problem 1** Given a vector $x \in \mathbb{R}^m$, $x \neq 0$, find an orthogonal matrix $M \in \mathbb{R}^{m \times m}$ such that $M^T x = \|x\|_2 \cdot e_1$, where $e_1 = (1, 0, \dots, 0)^T$ denotes the first unit vector.

In the rest of this section, we will first discuss how Householder reflections and Givens rotations can be used for solving Problem 1. Next, using these elementary

results, we show how one can construct the QR decomposition. Finally, we briefly mention the Gram–Schmidt orthogonalization method, which also provides a way to find the QR decomposition.

Householder Reflections

The QR decomposition using Householder reflections (HR) was developed by Golub (1965). Householder reflection (or Householder transformation) is a matrix P ,

$$P = I - \frac{1}{c} \mathbf{u} \mathbf{u}^\top, \quad c = \frac{1}{2} \mathbf{u}^\top \mathbf{u}, \quad (4.3)$$

where \mathbf{u} is a Householder vector. By definition, the matrix P is orthonormal and symmetric. Moreover, for any $\mathbf{x} \in \mathbb{R}^m$, it holds that

$$P\mathbf{x} = \mathbf{x} - \frac{1}{c} (\mathbf{u}^\top \mathbf{x}) \mathbf{u}.$$

Therefore, to apply HR one does not need to explicitly compute the matrix P itself. Additionally, it holds $P\mathbf{u} = -\mathbf{u}$ and $P\mathbf{x} \in \text{span}\{\mathbf{x}, \mathbf{u}\}$. This means that HR reflects a vector \mathbf{x} with respect to the hyperplane with normal vector \mathbf{u} (hence the name Householder reflection).

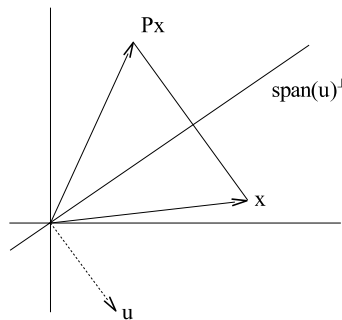


Figure 4.2. Reflection with respect to the hyperplane with a normal vector \mathbf{u}

To solve Problem 1 using some HR, we search for a vector \mathbf{u} such that \mathbf{x} will be reflected to the x -axis. This holds for the following choice of \mathbf{u} :

$$\mathbf{u} = \mathbf{x} + s_1 \|\mathbf{x}\|_2 \cdot \mathbf{e}_1, \quad s_1 = 2I(x_1 \geq 0) - 1, \quad (4.4)$$

where $x_1 = \mathbf{x}^\top \mathbf{e}_1$ denotes the first element of the vector \mathbf{x} and $I(\cdot)$ represents an indicator. For this reflection, it holds that c from (4.3) equals $\|\mathbf{x}\|_2(\|\mathbf{x}\|_2 + |x_1|)$ and $P\mathbf{x} = -s_1 \|\mathbf{x}\|_2 \cdot \mathbf{e}_1$ as one can verify by substituting (4.4) into (4.3).

Givens Rotations

A Givens rotation (GR) in m dimensions (or Givens transformation) is defined by an orthonormal matrix $R_{ij}(\alpha) \in \mathbb{R}^{m \times m}$,

$$R_{ij}(\alpha) = \begin{pmatrix} 1 & 0 & \cdots & \cdots & \cdots & \cdots & 0 \\ 0 & \ddots & & & & & \vdots \\ \vdots & & c & s & & & \vdots \\ \vdots & & & \ddots & & & \vdots \\ \vdots & & -s & c & & & \vdots \\ \vdots & & & & \ddots & & 0 \\ 0 & \cdots & \cdots & \cdots & \cdots & \cdots & 0 & 1 \end{pmatrix} \quad (4.5)$$

where $c = \cos \alpha$ and $s = \sin \alpha$ for $\alpha \in \mathbb{R}$ and $1 \leq i < j \leq n$. Thus, the rotation $R_{ij}(\alpha)$ represents a plane rotation in the space spanned by the unit vectors e_i and e_j by an angle α . In two dimensions, rotation $R_{12}(\alpha)$,

$$R_{12}(\alpha) = \begin{pmatrix} c & s \\ -s & c \end{pmatrix}, \quad c = \cos \alpha, \quad s = \sin \alpha$$

represents a clockwise rotation by an angle α ; see Fig. 4.3.

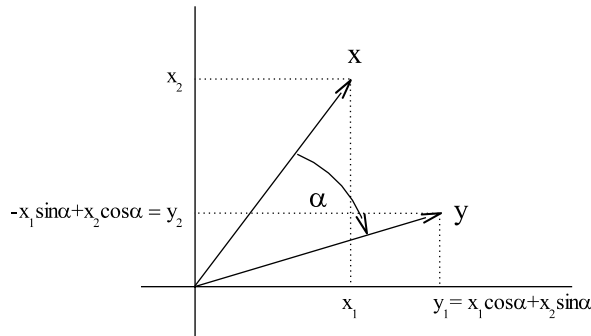


Figure 4.3. Rotation of x in a plane by an angle α

Now, let us have a look at how GRs can be used for solving Problem 1. A GR of a vector $\mathbf{x} = (x_1, \dots, x_m)^T \in \mathbb{R}^m$ by an angle α results in $R_{ij}(\alpha)\mathbf{x} = \mathbf{y} = (y_1, \dots, y_m)^T$ such that

$$y_k = \begin{cases} x_k & \text{for } k \neq i, j, \\ cx_i + sx_j & \text{for } k = i, \\ -sx_i + cx_j & \text{for } k = j. \end{cases}$$

For a vector \mathbf{x} with nonzero elements x_i or x_j , setting $d = (x_i^2 + x_j^2)^{1/2}$, $c = x_i/d$, $s = x_j/d$ leads to

$$\begin{pmatrix} c & s \\ -s & c \end{pmatrix} \begin{pmatrix} x_i \\ x_j \end{pmatrix} = \begin{pmatrix} d \\ 0 \end{pmatrix}.$$

Thus, using GR with this specific choice of c and s (referred further as \mathbf{R}_{ij}^0) implies that the j th component of the vector \mathbf{x} vanishes. Similarly to HRs, it is not necessary to explicitly construct the whole matrix \mathbf{P} to transform \mathbf{x} since the rotation is fully described by only two numbers: c and s . This elementary rotation \mathbf{R}_{ij}^0 does not however constitute a solution to Problem 1 yet: we need to combine more of them.

The next step employs a simple fact that the pre- or postmultiplication of a vector \mathbf{x} or a matrix \mathbf{A} by any GR $\mathbf{R}_{ij}(\alpha)$ affects only the i th and j th rows and columns, respectively. Hence, one can combine several rotations without one rotation spoiling the result of another rotation. (Consequently, GRs are more flexible than HRs). Two typical ways how GRs are used for solving Problem 1 mentioned in Sect. 4.1.3 follow.

1. $\mathbf{R}_{1n}^0 \mathbf{R}_{1,n-1}^0 \dots \mathbf{R}_{13}^0 \mathbf{R}_{12}^0 \mathbf{x} = d\mathbf{e}_1$. Here the k th component of the vector \mathbf{x} vanishes after the Givens rotation \mathbf{R}_{1k}^0 . The previously zeroed elements x_2, \dots, x_{k-1} are not changed because rotation \mathbf{R}_{1k} affects only the first and k th component.
2. $\mathbf{R}_{12}^0 \mathbf{R}_{23}^0 \dots \mathbf{R}_{n-1,n}^0 \mathbf{x} = d\mathbf{e}_1$. Here the k th component vanishes by the rotation $\mathbf{R}_{k-1,k}$.

Finally, there are several algorithms for computing the Givens rotations that improve over the straightforward evaluation of $\mathbf{R}_{ij}^0 \mathbf{x}$. A robust algorithm minimizing the loss of precision is given in Björck (1996). An algorithm minimizing memory requirements was proposed by Stewart (1976). On the other hand, Gentleman (1973) and Hammarling (1974) proposed modifications aiming to minimize the number of arithmetic operations.

QR Decomposition by Householder Reflections or Givens Rotations

An appropriate combination of HRs or GRs, respectively, can be used to compute the QR decomposition of a given matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$, $m \geq n$, in a following way. Let \mathbf{Q}_i , $i = 1, \dots, n-1$, denote an orthonormal matrix in $\mathbb{R}^{m \times m}$ such that premultiplication of $\mathbf{B} = \mathbf{Q}_{i-1} \dots \mathbf{Q}_1 \mathbf{A}$ by \mathbf{Q}_i can zero all elements in the i th column that are below the diagonal and such that the previous columns $1, \dots, i-1$ are not affected at all. Such a matrix can be a blockwise diagonal matrix with blocks being the identity matrix \mathbf{I}_{i-1} and a matrix \mathbf{M} solving Problem 1 for the vector composed of elements in the i th column of \mathbf{B} that lie on and below the diagonal. The first part \mathbf{I}_{i-1} guarantees that the columns $1, \dots, i-1$ of matrix \mathbf{B} are not affected by multiplication, whereas the second block \mathbf{M} transforms all elements in the i th column that are below the diagonal to zero. Naturally, matrix \mathbf{M} can be found by means of HRs or GRs as described in previous paragraphs.

This way, we construct a series of matrices Q_1, \dots, Q_n such that

$$Q_n \cdots Q_1 A = \begin{pmatrix} R \\ \mathbf{0} \end{pmatrix}.$$

Since all matrices Q_1, \dots, Q_n are orthonormal, $Q_t = Q_n \cdots Q_1$ is also orthonormal and its inverse equals its transpose: $Q_t^{-1} = Q_t^\top$. Hence,

$$A = (Q_n \cdots Q_1)^\top \begin{pmatrix} R \\ \mathbf{0} \end{pmatrix} = Q \begin{pmatrix} R \\ \mathbf{0} \end{pmatrix}$$

as described in Theorem 3.

We describe now the QR algorithm using HRs or GRs. Let $M(\mathbf{x})$ denote the orthonormal matrix from Problem 1 constructed for a vector \mathbf{x} by one of the discussed methods.

Algorithm 3

```

 $Q = I_m$ 
 $R = A$ 
for  $i = 1$  to  $n$ 
   $\mathbf{x} = \{R_{ki}\}_{k=i}^m$ 
   $Q_i = \begin{pmatrix} I_{i-1} & \mathbf{0} \\ \mathbf{0} & M(\mathbf{x}) \end{pmatrix}$ 
   $Q = Q_i Q$ 
   $R = Q_i R$ 
end
 $Q = Q^\top$ 
 $R = \{R_{ij}\}_{i=1, j=1}^{n, n}$ 

```

There are also modifications of this basic algorithm employing pivoting for better numerical performance and even revealing rank of the system, see Hong and Tan (1992) and Higham (2000) for instance. An error analysis of the QR decomposition by HRs and GRs are given by Gentleman (1975) and Higham (2000), respectively.

Gram–Schmidt Orthogonalization

Given a nonsingular matrix $A \in \mathbb{R}^{m \times n}$, $m \geq n$, the Gram–Schmidt orthogonalization constructs a matrix Q such that the columns of Q are orthonormal to each other and span the same space as the columns of A . Thus, A can be expressed as Q multiplied by another matrix R , whereby the Gram–Schmidt orthogonalization process (GS) ensures that R is an upper triangular matrix. Consequently, GS can be used to construct the QR decomposition of a matrix A . A survey of GS variants and their properties is given by Björck (1994).

The classical Gram–Schmidt (CGS) process constructs the orthonormal basis stepwise. The first column Q_1 of Q is simply normalized A_1 . Having constructed an orthonormal base $Q_{1:k} = \{Q_1, \dots, Q_k\}$, the next column Q_{k+1} is proportional to A_{k+1} minus its projection to the space $\text{span}\{Q_{1:k}\}$. Thus, Q_{k+1} is by its definition orthogonal to $\text{span}\{Q_{1:k}\}$, and at the same time, the first k columns of A and Q span the same linear space. The elements of the triangular matrix R from Theorem 3 are then coordinates of the columns of A given the columns of Q as a basis.

Algorithm 4

```

for i = 1 to n
  for j = 1 to i - 1
     $R_{ji} = Q_j^\top A_i$ 
  end
   $Q_i = A_i - \sum_{j=1}^{i-1} R_{ji} Q_j$ 
   $R_{ii} = (Q_i^\top Q_i)^{1/2}$ 
   $Q_i = Q_i / R_{ii}$ 
end

```

Similarly to many decomposition algorithms, also CGS allows a memory efficient implementation since the computed orthonormal columns of Q can rewrite the original columns of A . Despite this feature and mathematical correctness, the CGS algorithm does not always behave well numerically because numerical errors can very quickly accumulate. For example, an error made in computing Q_1 affects Q_2 , errors in both of these terms (although caused initially just by an error in Q_1) adversely influence Q_3 and so on. Fortunately, there is a modified Gram–Schmidt (MGS) procedure, which prevents such an error accumulation by subtracting linear combinations of Q_k directly from A before constructing following orthonormal vectors. (Surprisingly, MGS is historically older than CGS.)

Algorithm 5

```

for i = 1 to n
   $Q_i = A_i$ 
   $R_{ii} = (Q_i^\top Q_i)^{1/2}$ 
   $Q_i = Q_i / R_{ii}$ 
  for j = i + 1 to n
     $R_{ji} = Q_i^\top A_j$ 
     $A_j = A_j - R_{ji} Q_i$ 
  end
end

```

Apart from this algorithm (the row version of MGS), there are also a column version of MGS by Björck (1994) and MGS modifications employing iterative

orthogonalization and pivoting by Dax (2000). Numerical superiority of MGS over CGS was experimentally established already by Rice (1966). This result is also theoretically supported by the GS error analysis in Björck (1994), who uncovered numerical equivalence of the QR decompositions done by MGS and HRs.

4.1.4 Singular Value Decomposition

The singular value decomposition (SVD) plays an important role in numerical linear algebra and in many statistical techniques as well. Using two orthonormal matrices, SVD can diagonalize any matrix A and the results of SVD can tell a lot about (numerical) properties of the matrix. (This is closely related to the eigenvalue decomposition: any symmetric square matrix A can be diagonalized, $A = VDV^T$, where D is a diagonal matrix containing the eigenvalues of A and V is an orthonormal matrix.)

5 Theorem 5 Let $A \in \mathbb{R}^{m \times n}$ be a matrix of rank r . Then there exist orthonormal matrices $U \in \mathbb{R}^{m \times m}$ and $V \in \mathbb{R}^{n \times n}$ and a diagonal matrix $D \in \mathbb{R}^{m \times n}$, with the diagonal elements $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_r > \sigma_{r+1} = \dots = \sigma_{\min\{m,n\}} = 0$, such that $A = UDV^T$.

Numbers $\sigma_1, \dots, \sigma_{\min\{m,n\}}$ represent the singular values of A . Columns U_i and V_i of matrices U and V are called the left and right singular vectors of A associated with singular value σ_i , respectively, because $AV_i = \sigma_i U_i$ and $U_i^T A = \sigma_i V_i^T$, $i = 1, \dots, \min\{m, n\}$.

Similarly to the QR decomposition, SVD offers a numerically stable way to solve a system of linear equations. Given a system $Ax = UDV^T x = b$, one can transform it to $U^T Ax = DV^T x = U^T b$ and solve it in two trivial steps: first, finding a solution z of $Dz = U^T b$, and second, setting $x = Vz$, which is equivalent to $V^T x = z$.

On the other hand, the power of SVD lies in its relation to many important matrix properties; see Trefethen and Bau (1997), for instance. First of all, the singular values of a matrix A are equal to the (positive) square roots of the eigenvalues of $A^T A$ and AA^T , whereby the associated left and right singular vectors are identical with the corresponding eigenvectors. Thus, one can compute the eigenvalues of $A^T A$ directly from the original matrix A . Second, the number of nonzero singular values equals the rank of a matrix. Consequently, SVD can be used to find an effective rank of a matrix, to check a near singularity and to compute the condition number of a matrix. That is, it allows to assess conditioning and sensitivity to errors of a given system of equations. Finally, let us note that there are far more uses of SVD: identification of the null space of A , $\text{null}(A) = \text{span}\{V_{k+1}, \dots, V_n\}$; computation of the matrix pseudo-inverse, $A^- = VD^-U^T$; low-rank approximations and so on. See Björck (1996) and Trefethen and Bau (1997) for details.

Let us now present an overview of algorithms for computing the SVD decomposition, which are not described in details due to their extent. The first stable

algorithm for computing the SVD was suggested by Golub and Kahan (1965). It involved reduction of a matrix A to its bidiagonal form by HRs, with singular values and vectors being computed as eigenvalues and eigenvectors of a specific tridiagonal matrix using a method based on Sturm sequences. The final form of the QR algorithm for computing SVD, which has been the preferred SVD method for dense matrices up to now, is due to Golub and Reinsch (1970); see Anderson et al. (1999), Björck (1996) or Gentle (1998) for the description of the algorithm and some modifications. An alternative approach based on Jacobi algorithm was given by Hari and Veselić (1987). Latest contributions to the pool of computational methods for SVD, including von Matt (1995), Demmel et al. (1999) and Higham (2000), aim to improve the accuracy of singular values and computational speed using recent advances in the QR decomposition.

Matrix Inversion

4.1.5

In previous sections, we described how matrix decompositions can be used for solving systems of linear equations. Let us now discuss the use of matrix decompositions for inverting a nonsingular squared matrix $A \in \mathbb{R}^{n \times n}$, although matrix inversion is not needed very often. All discussed matrix decomposition construct two or more matrices A_1, \dots, A_d such that $A = A_1 \cdot \dots \cdot A_d$, where matrices $A_l, l = 1, \dots, d$, are orthonormal, triangular, or diagonal. Because $A^{-1} = A_d^{-1} \cdot \dots \cdot A_1^{-1}$, we just need to be able to invert orthonormal and triangular matrices (a diagonal matrix is a special case of a triangular matrix).

First, an orthonormal matrix Q satisfies by definition $Q^T Q = Q Q^T = I_n$. Thus, inversion is in this case equivalent to the transposition of a matrix: $Q^{-1} = Q^T$.

Second, inverting an upper triangular matrix U can be done by solving directly $XU = I_n$, which leads to the backward substitution method. Let $X = \{X_{ij}\}_{i=1, j=1}^{n, n}$ denote the searched for inverse matrix U^{-1} .

Algorithm 6

```

 $X = \mathbf{0}_n$ 
for i = n to 1
   $X_{ii} = 1/U_{ii}$ 
  for j = i + 1 to n
     $X_{ij} = -\left(\sum_{k=i+1}^j X_{kj}U_{ik}\right)/U_{ij}$ 
  end
end
end
```

The inversion of a lower triangular matrix L can be done analogously: the algorithm is applied to L^T , that is, U_{ij} is replaced by L_{ji} for $i, j = 1, \dots, n$.

There are several other algorithms available such as forward substitution or blockwise inversion. Designed for a faster and more (time) efficient computation, their numerical behavior does not significantly differ from the presented algorithm. See Croz and Higham (1992) for an overview and numerical study.

Direct Methods for Solving Linear Systems

A system of linear equations can be written in the matrix notation as

$$Ax = b, \quad (4.6)$$

where A denotes the coefficient matrix, b is the right-hand side, and x represents the solution vector we search for. The system (4.6) has a solution if and only if b belongs to the vector space spanned by the columns of A .

- If $m < n$, that is, the number of equations is smaller than the number of unknown variables, or if $m \geq n$ but A does not have a full rank (which means that some equations are linear combinations of the other ones), the system is underdetermined and there are either no solution at all or infinitely many of them. In the latter case, any solution can be written as a sum of a particular solution and a vector from the nullspace of A . Finding the solution space can involve the SVD decomposition (Sect. 4.1.4).
- If $m > n$ and the matrix A has a full rank, that is, if the number of equations is greater than the number of unknown variables, there is generally no solution and the system is overdetermined. One can search some x such that the distance between Ax and b is minimized, which leads to the linear least-squares problem if distance is measured by L_2 norm; see Chap. III.8.
- If $m = n$ and the matrix A is nonsingular, the system (4.6) has a unique solution. Methods suitable for this case will be discussed in the rest of this section as well as in Sect. 4.3.

From here on, we concentrate on systems of equations with unique solutions.

There are two basic classes of methods for solving system (4.6). The first class is represented by direct methods. They theoretically give an exact solution in a (predictable) finite number of steps. Unfortunately, this does not have to be true in computational praxis due to rounding errors: an error made in one step spreads in all following steps. Classical direct methods are discussed in this section. Moreover, solving an equation system by means of matrix decompositions, as discussed in Sect. 4.1, can be classified as a direct method as well. The second class is called iterative methods, which construct a series of solution approximations that (under some assumptions) converges to the solution of the system. Iterative methods are discussed in Sect. 4.3. Finally, note that some methods are on the borderline between the two classes; for example, gradient methods (Sect. 4.3.5) and iterative refinement (Sect. 4.2.2).

Further, the direct methods discussed in this section are not necessarily optimal for an arbitrary system (4.6). Let us deal with the main exceptions. First, even if a unique solution exist, numerical methods can fail to find the solution: if the number of unknown variables n is large, rounding errors can accumulate and result in a wrong solution. The same applies very much to sys-

tems with a nearly singular coefficient matrix. One alternative is to use iterative methods (Sect. 4.3), which are less sensitive to these problems. Another approach is to use the QR or SVD decompositions (Sect. 4.1), which can transform some nearly singular problems to nonsingular ones. Second, very large problems including hundreds or thousands of equations and unknown variables may be very time demanding to solve by standard direct methods. On the other hand, their coefficient matrices are often sparse, that is, most of their elements are zeros. Special strategies to store and solve such problems are discussed in Sect. 4.5.

To conclude these remarks, let us mention a close relation between solving the system (4.6) and computing the inverse matrix A^{-1} :

- having an algorithm that for a matrix A computes A^{-1} , we can find the solution to (4.6) as $\mathbf{x} = A^{-1}\mathbf{b}$;
- an algorithm solving the system (4.6) can be used to compute A^{-1} as follows. Solve n linear systems $A\mathbf{x}_i = \mathbf{e}_i$, $i = 1, \dots, n$ (or the corresponding system with multiple right-hand sides), where \mathbf{e}_i denotes the i th unit vector. Then $A^{-1} = (\mathbf{x}_1, \dots, \mathbf{x}_n)$.

In the rest of this section, we concentrate on the Gauss–Jordan elimination (Sect. 4.2.1) and its modifications and extensions, such as iterative refinement (Sect. 4.2.2). A wealth of information on direct methods can be found in monographs Axelsson (1994), Gentle (1998) and Golub and van Loan (1996).

Gauss–Jordan Elimination

4.2.1

In this subsection, we will simultaneously solve the linear systems

$$A\mathbf{x}_1 = \mathbf{b}_1, \quad A\mathbf{x}_2 = \mathbf{b}_2, \quad \dots, \quad A\mathbf{x}_k = \mathbf{b}_k$$

and a matrix equation $AX = B$, where $X, B \in \mathbb{R}^{n \times l}$ (its solution is $X = A^{-1}B$, yielding the inverse A^{-1} for a special choice $B = I_n$). They can be written as a linear matrix equation

$$A[\mathbf{x}_1|\mathbf{x}_2|\dots|\mathbf{x}_k|X] = [\mathbf{b}_1|\mathbf{b}_2|\dots|\mathbf{b}_k|B], \tag{4.7}$$

where the operator $|$ stands for column augmentation.

The Gauss–Jordan elimination (GJ) is based on elementary operations that do not affect the solution of an equation system. The solution of (4.7) will not change if we perform any of the following operations:

- interchanging any two rows of A and the corresponding rows of \mathbf{b}_i 's and B , $i = 1, \dots, k$;
- multiplying a row of A and the same row of \mathbf{b}_i 's and B by a nonzero number, $i = 1, \dots, k$;
- adding to a chosen row of A and the same row of \mathbf{b}_i 's and B a linear combination of other rows, $i = 1, \dots, k$.

Interchanging any two columns of A is possible too, but it has to be followed by interchanging the corresponding rows of all solutions x_i and X as well as of right sides b_i and B , $i = 1, \dots, k$. Each row or column operation described above is equivalent to the pre- or postmultiplication of the system by a certain elementary matrix R or C , respectively, that are results of the same operation applied to the identity matrix I_n .

GJ is a technique that applies one or more of these elementary operations to (4.7) so that A becomes the identity matrix I_n . Simultaneously, the right-hand side becomes the set of solutions. Denoting R_i , $i = 1, \dots, O$, the matrices corresponding to the i th row operation, the combination of all operations has to constitute inverse $A^{-1} = R_O \cdot \dots \cdot R_3 R_2 R_1$ and hence $x = R_O \cdot \dots \cdot R_3 R_2 R_1 b$. The exact choice of these elementary operation is described in the following paragraph.

Pivoting in Gauss–Jordan Elimination

Let us now discuss several well-known variants of the Gauss–Jordan elimination. GJ without pivoting does not interchange any rows or columns; only multiplication and addition of rows are permitted. First, nonzero nondiagonal elements in the first column A_1 are eliminated: the first row of (4.7) is divided by its diagonal element A_{11} and the A_{i1} -multiple of the modified first row is subtracted from the i th row, $i = 2, \dots, n$. We can proceed the same way for all n columns of A , and thus, transform A to the identity matrix I_n . It is easy to see that the method fails if the diagonal element in a column to be eliminated, the so-called pivot, is zero in some step. Even if this is not the case, one should be aware that GJ without pivoting is numerically unstable.

On the other hand, the GJ method becomes stable when using pivoting. This means that one can interchange rows (partial pivoting) or rows and columns (full pivoting) to put a suitable matrix element to the position of the current pivot. Since it is desirable to keep the already constructed part of the identity matrix, only rows below and columns right to the current pivot are considered. GJ with full pivoting is numerically stable. From the application point of view, GJ with partial pivoting is numerically stable too, although there are artificial examples where it fails. Additionally, the advantage of partial pivoting (compared to full pivoting) is that it does not change the order of solution components.

There are various strategies to choose a pivot. A very good choice is the largest available element (in absolute value). This procedure depends however on the original scaling of the equations. Implicit pivoting takes scaling into account and chooses a pivot as if the original system were rescaled so that the largest element of each row would be equal to one.

Finally, let us add several concluding remarks on efficiency of GJ and its relationship to matrix decompositions. As shown, GJ can efficiently solve problems with multiple right-hand sides known in advance and compute A^{-1} at the same time. On the other hand, if it is necessary to solve later a new system with the

same coefficient matrix A but a new right-hand side \mathbf{b} , one has to start the whole elimination process again, which is time demanding, or compute $A^{-1}\mathbf{b}$ using the previously computed inverse matrix A^{-1} , which leads to further error accumulation. In praxis, one should prefer matrix decompositions, which do not have this drawback. Specifically, the LU decomposition (Sect. 4.1.2) is equivalent to GJ (with the same kind of pivoting applied in both cases) and allows us to repeatedly solve systems with the same coefficient matrix in an efficient way.

Iterative Refinement

4.2.2

In the introduction to Sect. 4.2, we noted that direct methods are rather sensitive to rounding errors. Iterative refinement offers a way to improve the solution obtained by any direct method, unless the system matrix A is too ill-conditioned or even singular.

Let \mathbf{x}_1 denote an initially computed (approximate) solution of (4.6). Iterative refinement is a process constructing a series \mathbf{x}_i , $i = 1, 2, \dots$, as described in Algorithm 7. First, given a solution \mathbf{x}_i , the residuum $\mathbf{r}_i = A\mathbf{x}_i - \mathbf{b}$ is computed. Then, one obtains the correction $\Delta\mathbf{x}_i$ by solving the original system with residuum \mathbf{r}_i on the right-hand side.

Algorithm 7

```
Repeat for  $i = 1, 2, \dots$ 
  compute  $\mathbf{r}_i = \mathbf{b} - A\mathbf{x}_i$ 
  solve  $A\Delta\mathbf{x}_i = \mathbf{r}_i$  for  $\Delta\mathbf{x}_i$ 
  set  $\mathbf{x}_{i+1} = \mathbf{x}_i + \Delta\mathbf{x}_i$ 
until the desired precision is achieved.
```

It is reasonable to carry out the computation of residuals \mathbf{r}_i in a higher precision because a lot of cancellation occurs if \mathbf{x}_i is a good approximation. Nevertheless, provided that the coefficient matrix A is not too ill-conditioned, Skeel (1980) proved that GJ with partial pivoting and only one step of iterative refinement computed in a fixed precision is stable (it has a relative backward error proportional to the used precision). In spite of this result, one can recommend to use iterative refinement repeatedly until the desired precision is reached.

Additionally, an important feature of iterative refinement is its low computational costs. Provided that a system is solved by means of decompositions (e.g., GJ is implemented as the LU decomposition), a factorization of A is available already after computing the initial solution \mathbf{x}_1 . Subsequently, solving any system with the same coefficient matrix A , such as $A\Delta\mathbf{x}_i = \mathbf{r}_i$, can be done fast and efficiently and the computational costs of iterative refinement are small.

Iterative Methods for Solving Linear Systems

4.3

Direct methods for solving linear systems theoretically give the exact solution in a finite number of steps, see Sect. 4.2. Unfortunately, this is rarely true in applications because of rounding errors: an error made in one step spreads further in all following steps! Contrary to direct methods, iterative methods construct a series of solution approximations such that it converges to the exact solution of a system. Their main advantage is that they are self-correcting, see Sect. 4.3.1.

In this section, we first discuss general principles of iterative methods that solve linear system (4.6), $A\mathbf{x} = \mathbf{b}$, whereby we assume that $A \in \mathbb{R}^{n \times n}$ and the system has exactly one solution \mathbf{x}_e (see Sect. 4.2 for more details on other cases). Later, we describe most common iterative methods: the Jacobi, Gauss–Seidel, successive overrelaxation, and gradient methods (Sects. 4.3.2–4.3.5). Monographs containing detailed discussion of these methods include Björck (1996), Golub and van Loan (1996) and Hackbusch (1994). Although we treat these methods separately from the direct methods, let us mention here that iterative methods can usually benefit from a combination with the Gauss elimination, see Milaszewicz (1987) and Alanelli and Hadjidimos (2004), for instance.

To unify the presentation of all methods, let D , L , and U denote the diagonal, lower triangular and upper triangular parts of a matrix A throughout this section:

$$D_{ij} = \begin{cases} A_{ij} & \text{for } i = j, \\ 0 & \text{otherwise;} \end{cases} \quad L_{ij} = \begin{cases} A_{ij} & \text{for } i > j, \\ 0 & \text{otherwise;} \end{cases} \quad U_{ij} = \begin{cases} A_{ij} & \text{for } i < j, \\ 0 & \text{otherwise.} \end{cases}$$

General Principle of Iterative Methods for Linear Systems

4.3.1

An iterative method for solving a linear system $A\mathbf{x} = \mathbf{b}$ constructs an iteration series \mathbf{x}_i , $i = 0, 1, 2, \dots$, that under some conditions converges to the exact solution \mathbf{x}_e of the system ($A\mathbf{x}_e = \mathbf{b}$). Thus, it is necessary to choose a starting point \mathbf{x}_0 and iteratively apply a rule that computes \mathbf{x}_{i+1} from an already known \mathbf{x}_i .

A starting vector \mathbf{x}_0 is usually chosen as some approximation of \mathbf{x} . (Luckily, its choice cannot cause divergence of a convergent method.) Next, given \mathbf{x}_i , $i \in \mathbb{N}$, the subsequent element of the series is computed using a rule of the form

$$\mathbf{x}_{i+1} = \mathbf{B}_i \mathbf{x}_i + \mathbf{C}_i \mathbf{b}, \quad i = 0, 1, 2, \dots, \quad (4.8)$$

where $\mathbf{B}_i, \mathbf{C}_i \in \mathbb{R}^{n \times n}$, $i \in \mathbb{N}$, are matrix series. Different choices of \mathbf{B}_i and \mathbf{C}_i define different iterative methods.

Let us discuss now a minimal set of conditions on B_i and C_i in (4.8) that guarantee the convergence of an iterative method. First of all, it has to hold that $B_i + C_i A = I_n$ for all $i \in \mathbb{N}$, or equivalently,

$$\mathbf{x}_e = B_i \mathbf{x}_e + C_i \mathbf{b} = (B_i + C_i A) \mathbf{x}_e, \quad i \in \mathbb{N}. \quad (4.9)$$

In other words, once the iterative process reaches the exact solution \mathbf{x}_e , all consecutive iterations should stay equal to \mathbf{x}_e and the method cannot depart from this solution. Second, starting from a point $\mathbf{x}_0 \neq \mathbf{x}_e$, we have to ensure that approximations \mathbf{x}_i will converge to \mathbf{x}_e as i increases.

Theorem 6 An iteration series \mathbf{x}_i given by (4.8) converges to the solution of system (4.6) for any chosen \mathbf{x}_0 iff **6**

$$\lim_{i \rightarrow \infty} B_i B_{i-1} \dots B_0 = \mathbf{0}.$$

In praxis, stationary iterative methods are used, that is, methods with constant $B_i = B$ and $C_i = C$, $i \in \mathbb{N}$. Consequently, an iteration series is then constructed using

$$\mathbf{x}_{i+1} = B \mathbf{x}_i + C \mathbf{b}, \quad i = 0, 1, 2, \dots \quad (4.10)$$

and the convergence condition in Theorem 6 has a simpler form.

Theorem 7 An iteration series \mathbf{x}_i given by (4.10) converges to the solution of system (4.6) for any chosen \mathbf{x}_0 iff the spectral radius $\rho(B) < 1$, where $\rho(B) = \max_{i=1, \dots, n} |\lambda_i|$ and $\lambda_1, \dots, \lambda_n$ represent the eigenvalues of B . **7**

Note that the convergence condition $\rho(B) < 1$ holds, for example, if $\|B\| < 1$ in any matrix norm. Moreover, Theorem 7 guarantees the self-correcting property of iterative methods since convergence takes place independent of the starting value \mathbf{x}_0 . Thus, if computational errors adversely affect \mathbf{x}_i during the i th iteration, \mathbf{x}_i can be considered as a new starting vector and the iterative method will further converge. Consequently, the iterative methods are in general more robust than the direct ones.

Apparently, such an iterative process can continue arbitrarily long unless $\mathbf{x}_i = \mathbf{x}_e$ at some point. This is impractical and usually unnecessary. Therefore, one uses stopping (or convergence) criteria that stop the iterative process when a pre-specified condition is met. Commonly used stopping criteria are based on the change of the solution or residual vector achieved during one iteration. Specifically, given a small $\varepsilon > 0$, the iterative process is stopped after the i th iteration when $\|\mathbf{x}_i - \mathbf{x}_{i-1}\| \leq \varepsilon$, $\|\mathbf{r}_i - \mathbf{r}_{i-1}\| \leq \varepsilon$, or $\|\mathbf{r}_i\| \leq \varepsilon$, where $\mathbf{r}_i = A \mathbf{x}_i - \mathbf{b}$ is a residual vector. Additionally, a maximum acceptable number of iterations is usually specified.

4.3.2 Jacobi Method

The Jacobi method is motivated by the following observation. Let A have nonzero diagonal elements (the rows of any nonsingular matrix can be reorganized to achieve this). Then the diagonal part D of A is nonsingular and the system (4.6) can be rewritten as $D\mathbf{x} + (\mathbf{L} + \mathbf{U})\mathbf{x} = \mathbf{b}$. Consequently,

$$\mathbf{x} = D^{-1}[(-\mathbf{L} - \mathbf{U})\mathbf{x} + \mathbf{b}].$$

Replacing \mathbf{x} on the left-hand side by \mathbf{x}_{i+1} and \mathbf{x} on the right-hand side by \mathbf{x}_i leads to the iteration formula of the Jacobi method:

$$\mathbf{x}_{i+1} = -D^{-1}(\mathbf{L} + \mathbf{U})\mathbf{x}_i + D^{-1}\mathbf{b}.$$

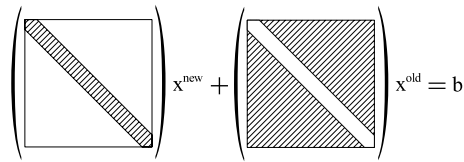


Figure 4.4. Scheme of the Jacobi method

The intuition of the Jacobi method is very simple: given an approximation \mathbf{x}^{old} of the solution, let us express the k th component x_k of \mathbf{x} as a function of the other components from the k th equation and compute x_k given \mathbf{x}^{old} :

$$x_k^{\text{new}} = \frac{1}{A_{kk}} \left(b_k - \sum_{\substack{j=1 \\ j \neq k}}^n A_{kj} x_j^{\text{old}} \right), \quad (4.11)$$

$k = 1, \dots, n$ (see Fig. 4.4).

The Jacobi method converges for any starting vector \mathbf{x}_0 as long as $\rho(D^{-1}(\mathbf{L} + \mathbf{U})) < 1$, see Theorem 7. This condition is satisfied for a relatively big class of matrices including diagonally dominant matrices (matrices A such that $\sum_{j=1, j \neq i}^n |A_{ij}| \leq |A_{ii}|$ for $i = 1, \dots, n$), and symmetric matrices A such that $D, A = L + D + U$, and $-\mathbf{L} + \mathbf{D} - \mathbf{U}$ are all positive definite. Although there are many improvements to the basic principle of the Jacobi method in terms of convergence to \mathbf{x}_e , see Sects. 4.3.3 and 4.3.4, its advantage is an easy and fast implementation (elements of a new iteration \mathbf{x}_i can be computed independently of each other).

4.3.3 Gauss–Seidel Method

Analogously to the Jacobi method, we can rewrite system (4.6) as $(\mathbf{L} + \mathbf{D})\mathbf{x} + \mathbf{U}\mathbf{x} = \mathbf{b}$, which further implies $\mathbf{x} = (\mathbf{L} + \mathbf{D})^{-1}[-\mathbf{U}\mathbf{x} + \mathbf{b}]$. This leads to the iteration formula of the Gauss–Seidel method:

$$\mathbf{x}_{i+1} = -(\mathbf{L} + \mathbf{D})^{-1}\mathbf{U}\mathbf{x}_i + (\mathbf{L} + \mathbf{D})^{-1}\mathbf{b}. \quad (4.12)$$

The main difference to the Jacobi methods lies in a more efficient use of (4.11). When computing the k th element x_k^{new} , the first $k - 1$ elements $x_1^{\text{new}}, \dots, x_{k-1}^{\text{new}}$ are already known (and presumably more precise than $x_1^{\text{old}}, \dots, x_{k-1}^{\text{old}}$). Thus, it is possible to use these new values instead of the old ones and speed up the convergence (see Fig. 4.5 for a scheme). Moreover, using this strategy, the newly computed elements of \mathbf{x}_{i+1} can directly overwrite the respective elements of \mathbf{x}_i and save memory this way.

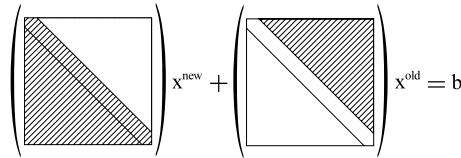


Figure 4.5. Scheme of the Gauss-Seidel method

Following the Theorem 7, the Gauss-Seidel method converges for any starting vector \mathbf{x}_0 if $\rho((L + D)^{-1}U) < 1$. This condition holds, for example, for diagonally dominant matrices as well as for positive definite ones.

Successive Overrelaxation Method

The successive overrelaxation (SOR) method aims to further refine the Gauss-Seidel method. The Gauss-Seidel formula (4.12) can be rewritten as

$$\mathbf{x}_{i+1} = \mathbf{x}_i - D^{-1}[\{L\mathbf{x}_{i+1} + (D + U)\mathbf{x}_i\} - \mathbf{b}] = \mathbf{x}_i - \Delta_i,$$

which describes the difference Δ_i between \mathbf{x}_{i+1} and \mathbf{x}_i expressed for the k th element of \mathbf{x}_{i+1} from the k th equation, $k = 1, \dots, n$. The question SOR poses is whether the method can converge faster if we “overly” correct \mathbf{x}_{i+1} in each step; that is, if \mathbf{x}_i is corrected by a multiple ω of Δ_i in each iteration. This idea leads to the SOR formula:

$$\mathbf{x}_{i+1} = \mathbf{x}_i - \omega D^{-1}[\{L\mathbf{x}_{i+1} + (D + U)\mathbf{x}_i\} - \mathbf{b}],$$

or in the form (4.10),

$$\mathbf{x}_{i+1} = (D + \omega L)^{-1}\{(1 - \omega)D - \omega U\}\mathbf{x}_i + \omega(D + \omega L)^{-1}\mathbf{b}. \tag{4.13}$$

The parameter ω is called the (over)relaxation parameter and it can be shown that SOR converges only for $\omega \in (0, 2)$, a result derived by Kahan (1958).

A good choice of parameter ω can speed up convergence, as measured by the spectral radius of the corresponding iteration matrix \mathbf{B} (see Theorem 7; a lower spectral radius $\rho(\mathbf{B})$ means faster convergence). There is a choice of literature devoted to the optimal setting of relaxation parameter: see Hadjidimos (2000) for a recent overview of the main results concerning SOR. We just present one important result, which is due to Young (1954).

1 **Definition 1** A matrix A is said to be two-cyclic consistently ordered if the eigenvalues of the matrix $M(\alpha) = \alpha D^{-1}L + \alpha^{-1}D^{-1}U$, $\alpha \neq 0$, are independent of α .

8 **Theorem 8** Let the matrix A be two-cyclic consistently ordered. Let the respective Gauss–Seidel iteration matrix $B = -(L + D)^{-1}U$ have the spectral radius $\rho(B) < 1$. Then the optimal relaxation parameter ω in SOR is given by

$$\omega_{\text{opt}} = \frac{2}{1 + \sqrt{1 - \rho(B)}}$$

and for this optimal value it holds $\rho(B; \omega_{\text{opt}}) = \omega_{\text{opt}} - 1$.

Using SOR with the optimal relaxation parameter significantly increases the rate of convergence. Note however that the convergence acceleration is obtained only for ω very close to ω_{opt} . If ω_{opt} cannot be computed exactly, it is better to take ω slightly larger rather than smaller. Golub and van Loan (1996) describe an approximation algorithm for $\rho(B)$.

On the other hand, if the assumptions of Theorem 8 are not satisfied, one can employ the symmetric SOR (SSOR), which performs the SOR iteration twice: once as usual, see (4.13), and once with interchanged L and U . SSOR requires more computations per iteration and usually converges slower, but it works for any positive definite matrix and can be combined with various acceleration techniques. See Björck (1996) and Hadjidimos (2000) for details.

4.3.5 Gradient Methods

Gradient iterative methods are based on the assumption that A is a symmetric positive definite matrix. They use this assumption to reformulate (4.6) as a minimization problem: \mathbf{x}_e is the only minimum of the quadratic form

$$Q(\mathbf{x}) = \frac{1}{2} \mathbf{x}^\top \mathbf{A} \mathbf{x} - \mathbf{x}^\top \mathbf{b}.$$

Given this minimization problem, gradient methods construct an iteration series of vectors converging to \mathbf{x}_e using the following principle. Having the i th approximation \mathbf{x}_i , choose a direction \mathbf{v}_i and find a number α_i such that the new vector

$$\mathbf{x}_{i+1} = \mathbf{x}_i + \alpha_i \mathbf{v}_i$$

is a minimum of $Q(\mathbf{x})$ on the line $\mathbf{x}_i + \alpha \mathbf{v}_i$, $\alpha \in \mathbb{R}$. Various choices of directions \mathbf{v}_i then render different gradient methods, which are in general nonstationary (\mathbf{v}_i changes in each iteration). We discuss here three methods: the Gauss–Seidel (as a gradient method), steepest descent and conjugate gradients methods.

Gauss–Seidel Method as a Gradient Method

Interestingly, the Gauss–Seidel method can be seen as a gradient method for the choice

$$\mathbf{v}_{kn+i} = \mathbf{e}_i, \quad k = 0, 1, 2, \dots, \quad i = 1, \dots, n,$$

where \mathbf{e}_i denotes the i th unit vector. The k th Gauss–Seidel iteration corresponds to n subiterations with \mathbf{v}_{kn+i} for $i = 1, \dots, n$.

Steepest Descent Method

The steepest descent method is based on the direction \mathbf{v}_i given by the gradient of $Q(\mathbf{x})$ at \mathbf{x}_i . Denoting the residuum of the i th approximation $\mathbf{r}_i = \mathbf{b} - \mathbf{A}\mathbf{x}_i$, the iteration formula is

$$\mathbf{x}_{i+1} = \mathbf{x}_i + \frac{\mathbf{r}_i^\top \mathbf{r}_i}{\mathbf{r}_i^\top \mathbf{A} \mathbf{r}_i} \mathbf{r}_i,$$

where \mathbf{r}_i represents the direction \mathbf{v}_i and its coefficient is the $Q(\mathbf{x})$ -minimizing choice of α_i . By definition, this method reduces $Q(\mathbf{x}_i)$ at each step, but it is not very effective. The conjugate gradient method discussed in the next subsection will usually perform better.

Conjugate Gradient Method

In the conjugate gradient (CG) method proposed by Hestenes and Stiefel (1952), the directions \mathbf{v}_i are generated by the \mathbf{A} -orthogonalization of residuum vectors. Given a symmetric positive definite matrix \mathbf{A} , \mathbf{A} -orthogonalization is a procedure that constructs a series of linearly independent vectors \mathbf{v}_i such that $\mathbf{v}_i^\top \mathbf{A} \mathbf{v}_j = 0$ for $i \neq j$ (conjugacy or \mathbf{A} -orthogonality condition). It can be used to solve the system (4.6) as follows ($\mathbf{r}_i = \mathbf{b} - \mathbf{A}\mathbf{x}_i$ represents residuals).

Algorithm 8

```

 $\mathbf{v}_0 = \mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0$ 
do
   $\alpha_i = (\mathbf{v}_i^\top \mathbf{r}_i) / (\mathbf{v}_i^\top \mathbf{A} \mathbf{v}_i)$ 
   $\mathbf{x}_{i+1} = \mathbf{x}_i + \alpha_i \mathbf{v}_i$ 
   $\mathbf{r}_{i+1} = \mathbf{r}_i - \alpha_i \mathbf{A} \mathbf{v}_i$ 
   $\beta_i = -(\mathbf{v}_i^\top \mathbf{A} \mathbf{r}_{i+1}) / (\mathbf{v}_i^\top \mathbf{A} \mathbf{v}_i)$ 
   $\mathbf{v}_{i+1} = \mathbf{r}_{i+1} + \beta_i \mathbf{v}_i$ 
until a stop criterion holds

```

An interesting theoretic property of CG is that it reaches the exact solution in at most n steps because there are not more than n (\mathbf{A} -)orthogonal vectors. Thus, CG is not a truly iterative method. (This does not have to be the case if \mathbf{A} is a singular or non-square matrix, see Kammerer and Nashed, 1972.) On the other hand, it is

usually used as an iterative method, because it can give a solution within the given accuracy much earlier than after n iterations. Moreover, if the approximate solution \mathbf{x}_n after n iterations is not accurate enough (due to computational errors), the algorithm can be restarted with \mathbf{x}_0 set to \mathbf{x}_n . Finally, let us note that CG is attractive for use with large sparse matrices because it addresses A only by its multiplication by a vector. This operation can be done very efficiently for a properly stored sparse matrix, see Sect. 4.5.

The principle of CG has many extensions that are applicable also for nonsymmetric nonsingular matrices: for example, generalized minimal residual, Saad and Schultz (1986); (stabilized) biconjugate gradients, Vorst (1992); or quasi-minimal residual, Freund and Nachtigal (1991).

4.4 Eigenvalues and Eigenvectors

In this section, we deal with methods for computing eigenvalues and eigenvectors of a matrix $A \in \mathbb{R}^{n \times n}$. First, we discuss a simple power method for computing one or few eigenvalues (Sect. 4.4.1). Next, we concentrate on methods performing the complete eigenanalysis, that is, finding all eigenvalues (the Jacobi, QR, and LR methods in Sects. 4.4.2–4.4.5). Finally, we briefly describe a way to improve already computed eigenvalues and to find the corresponding eigenvector. Additionally, note that eigenanalysis can be also done by means of SVD, see Sect. 4.1.4. For more details on the described as well as some other methods, one can consult monographs by Gentle (1998), Golub and van Loan (1996), Press et al. (1992) and Stoer and Bulirsch (2002).

Before discussing specific methods, let us describe the principle common to most of them. We assume that $A \in \mathbb{R}^{n \times n}$ has eigenvalues $|\lambda_1| \geq |\lambda_2| \geq \dots \geq |\lambda_n|$. To find all eigenvalues, we transform the original matrix A to a simpler matrix B such that it is similar to A (recall that matrices A and B are similar if there is a matrix T such that $B = T^{-1}AT$). The similarity of A and B is crucial since it guarantees that both matrices have the same eigenvalues and their eigenvectors follow simple relation: if \mathbf{g} is an eigenvector of B corresponding to its eigenvalue λ , then $T\mathbf{g}$ is an eigenvector of A corresponding to the same eigenvalue λ .

There are two basic strategies to construct a similarity transformation B of the original matrix A . First, one can use a series of simple transformations, such as GRs, and eliminate elements of A one by one (see the Jacobi method, Sect. 4.4.2). This approach is often used to transform A to its tridiagonal or upper Hessenberg forms. (Matrix B has the upper Hessenberg form if it is an upper triangular except for the first subdiagonal; that is, $A_{ij} = 0$ for $i > j + 1$, where $i, j = 1, \dots, n$). Second, one can also factorize A into $A = F_L F_R$ and switch the order of factors, $B = F_R F_L$ (similarity of A and B follows from $B = F_R F_L = F_L^{-1} A F_L$). This is used for example by the LR method (Sect. 4.4.5). Finally, there are methods combining both approaches.

Power Method

4.4.1

In its basic form, the power method aims at finding only the largest eigenvalue λ_1 of a matrix A and the corresponding eigenvector. Let us assume that the matrix A has a dominant eigenvalue ($|\lambda_1| > |\lambda_2|$) and n linearly independent eigenvectors.

The power method constructs two series c_i and \mathbf{x}_i , $i \in \mathbb{N}$, that converge to λ_1 and to the corresponding eigenvector \mathbf{g}_1 , respectively. Starting from a vector \mathbf{x}_0 that is not orthogonal to \mathbf{g}_1 , one only has to iteratively compute $A\mathbf{x}_i$ and split it to its norm c_{i+1} and the normalized vector \mathbf{x}_{i+1} , see Algorithm 9. Usually, the Euclidian ($c_{i+1} = \|A\mathbf{x}_i\|_2$) and maximum ($c_{i+1} = \max_{j=1,\dots,n} |(A\mathbf{x}_i)_j|$) norms are used.

Algorithm 9

```

i = 0
do
  i = i + 1
   $\mathbf{x}_{i+1} = A\mathbf{x}_i$ 
   $c_{i+1} = \|A\mathbf{x}_{i+1}\|$ 
   $\mathbf{x}_{i+1} = \mathbf{x}_{i+1}/c_{i+1}$ 
until a stop criterion holds

```

Although assessing the validity of assumptions is far from trivial, one can usually easily recognize whether the method converges from the behaviour of the two constructed series.

Furthermore, the power method can be extended to search also for other eigenvalues; for example, the smallest one and the second largest one. First, if A is nonsingular, we can apply the power method to A^{-1} to find the smallest eigenvalue λ_n because $1/\lambda_n$ is the largest eigenvalue of A^{-1} . Second, if we need more eigenvalues and λ_1 is already known, we can use a reduction method to construct a matrix B that has the same eigenvalues and eigenvectors as A except for λ_1 , which is replaced by zero eigenvalue. To do so, we need to find a (normalized) eigenvector \mathbf{h}_1 of A^\top corresponding to λ_1 (A and A^\top have the same eigenvalues) and to set $B = A - \lambda_1 \mathbf{h}_1 \mathbf{h}_1^\top$. Naturally, this process can be repeated to find the third and further eigenvalues.

Finally, let us mention that the power method can be used also for some matrices without dominant eigenvalue (e.g., matrices with $\lambda_1 = \dots = \lambda_p$ for some $1 < p \leq n$). For further extensions of the power method see Sidi (1989), for instance.

Jacobi Method

4.4.2

For a symmetric matrix A , the Jacobi method constructs a series of orthogonal matrices R_i , $i \in \mathbb{N}$, such that the matrix $T_i = R_i^\top \dots R_1^\top A R_1 \dots R_i$ converges to a diagonal matrix D . Each matrix R_i is a GR matrix defined in (4.5), whereby the angle α is chosen so that one nonzero element $(T_i)_{jk}$ becomes zero in T_{i+1} . Formulas for computing R_i given the element (j, k) to be zeroed are described in Gentle (1998),

for instance. Once the matrix A is diagonalized this way, the diagonal of D contains the eigenvalues of A and the columns of matrix $R = R_1 \cdot \dots \cdot R_i$ represent the associated eigenvectors.

There are various strategies to choose an element (j, k) which will be zeroed in the next step. The classical Jacobi method chooses the largest off-diagonal element in absolute value and it is known to converge. (Since searching the maximal element is time consuming, various systematic schemes were developed, but their convergence cannot be often guaranteed.) Because the Jacobi method is relatively slow, other methods are usually preferred (e.g., the QR method). On the other hand, it has recently become interesting again because of its accuracy and easy parallelization (Higham, 1997; Zhou and Brent, 2003).

4.4.3 Givens and Householder Reductions

The Givens and Householder methods use a similar principle as the Jacobi method. A series of GRs or HRs, designed such that they form similarity transformations, is applied to a symmetric matrix A in order to transform it to a tridiagonal matrix. (A tridiagonal matrix is the Hessenberg form for symmetric matrices.) This tridiagonal matrix is then subject to one of the iterative methods, such as the QR or LR methods discussed in the following paragraphs. Formulas for Givens and Householder similarity transformations are given in Press et al. (1992), for instance.

4.4.4 QR Method

The QR method is one of the most frequently used methods for the complete eigenanalysis of a nonsymmetric matrix, despite the fact that its convergence is not ensured. A typical algorithm proceeds as follows. In the first step, the matrix A is transformed into a Hessenberg matrix using Givens or Householder similarity transformations (see Sects. 4.1.3 and 4.4.3). In the second step, this Hessenberg matrix is subject to the iterative process called chasing. In each iteration, similarity transformations, such as GRs, are first used to create nonzero entries in positions $(i+2, i)$, $(i+3, i)$ and $(i+3, i+1)$ for $i = 1$. Next, similarity transformations are repeatedly used to zero elements $(i+2, i)$ and $(i+3, i)$ and to move these “nonzeros” towards the lower right corner of the matrix (i.e., to elements $(i+2, i)$, $(i+3, i)$ and $(i+3, i+1)$ for $i = i+1$). As a result of chasing, one or two eigenvalues can be extracted. If $A_{n,n-1}$ becomes zero (or negligible) after chasing, element $A_{n,n}$ is an eigenvalue. Consequently, we can delete the n th row and column of the matrix and apply chasing to this smaller matrix to find another eigenvalue. Similarly, if $A_{n-1,n-2}$ becomes zero (or negligible), the two eigenvalues of the 2×2 submatrix in the lower right corner are eigenvalues of A . Subsequently, we can delete last two rows and columns and continue with the next iteration.

Since a more detailed description of the whole iterative process goes beyond the extent of this contribution, we refer a reader to Gentle (1998) for a shorter discussion and to Golub and van Loan (1996) and Press et al. (1992) for a more detailed discussion of the QR method.

LR Method

4.4.5

The LR method is based on a simple observation that decomposing a matrix A into $A = F_L F_R$ and multiplying the factors in the inverse order results in a matrix $B = F_R F_L$ similar to A . Using the LU decomposing (Sect. 4.1.2), the LR method constructs a matrix series A_i for $i \in \mathbb{N}$, where $A_1 = A$ and

$$A_i = L_i U_i \implies A_{i+1} = U_i L_i,$$

where L_i is a lower triangular matrix and U_i is an upper triangular matrix with ones on its diagonal. For a wide class of matrices, including symmetric positive definite matrices, A_i and L_i are proved to converge to the same lower triangular matrix L , whereby the eigenvalues of A form the diagonal of L and are ordered by the decreasing absolute value.

Inverse Iterations

4.4.6

The method of inverse iterations can be used to improve an approximation λ^* of an eigenvalue λ of a matrix A . The method is based on the fact that the eigenvector g associated with λ is also an eigenvector of $\tilde{A} = (A - \lambda^* I)^{-1}$ associated with the eigenvalue $\tilde{\lambda} = (\lambda - \lambda^*)^{-1}$. For an initial approximation λ^* close to λ , $\tilde{\lambda}$ is the dominant eigenvalue of \tilde{A} . Thus, it can be computed by the power method described in Sect. 4.4.1, whereby λ^* could be modified in each iteration in order to improve the approximation of λ .

This method is not very efficient without a good starting approximation, and therefore, it is not suitable for the complete eigenanalysis. On the other hand, the use of the power method makes it suitable for searching of the eigenvector g associated with λ . Thus, the method of inverse iterations often complements methods for complete eigenanalysis and serves then as a tool for eigenvector analysis. For this purpose, one does not have to perform the iterative improvement of initial λ^* : applying the power method on $\tilde{A} = (A - \lambda^* I)^{-1}$ suffices. See Ipsen (1997), Press et al. (1992) and Stoer and Bulirsch (2002) for more details.

Sparse Matrices

4.5

Numerical problems arising in some applications, such as seemingly unrelated regressions, spatial statistics, or support vector machines (Chap. III.15), are sparse: they often involve large matrices, which have only a small number of nonzero elements. (It is difficult to specify what exactly “small number” is.) From the practical point of view, a matrix is sparse if it has so many zero elements that it is worth to inspect their structure and use appropriate methods to save storage and the number of operations. Some sparse matrices show a regular pattern of nonzero elements (e.g., band matrices), while some exhibit a rather irregular pattern. In both cases, solving the respective problem efficiently means to store and operate

on only nonzero elements and to keep the “fill,” the number of newly generated nonzero elements, as small as possible.

In this section, we first discuss some of storage schemes for sparse matrices, which could indicate what types of problems can be effectively treated as sparse ones (Sect. 4.5.1). Later, we give examples of classical algorithms adopted for sparse matrices (Sect. 4.5.2). Monographs introducing a range of methods for sparse matrices include Duff et al. (1989), Hackbusch (1994) and Saad (2003).

4.5.1 Storage Schemes for Sparse Matrices

To save storage, only nonzero elements of a sparse vector or matrix should be stored. There are various storage schemes, which require approximately from two to five times the number of nonzero elements to store a vector or a matrix. Unfortunately, there is no standard scheme. We discuss here the widely used and sufficiently general compressed (row) storage for vectors and for general and banded matrices.

The compressed form of a vector \mathbf{x} consists of a triplet $(\mathbf{c}, \mathbf{i}, n_0)$, where \mathbf{c} is a vector containing nonzero elements of \mathbf{x} , \mathbf{i} is an integer vector containing the indices of elements stored in \mathbf{c} and n_0 specifies the number of nonzero elements. The stored elements are related to the original vector by formula $x_{\{i_j\}} = c_j$ for $j = 1, \dots, n_0$. To give an example, the vector $\mathbf{x} = (0, 0, 3, 0, -8, 1.5, 0, 0, 0, 16, 0)$ could be stored as

$$\mathbf{c} = (3, 1.5, -8, 16), \quad \mathbf{i} = (3, 6, 5, 10), \quad n_0 = 4.$$

Obviously, there is no need to store the elements in the original order. Therefore, adding new nonzero elements is easy. Operations involving more sparse vectors are simpler if we can directly access elements of one vector, that is, if one of the vectors is “uncompressed.” For example, computing the inner product $a = \mathbf{x}^\top \mathbf{y}$ of a sparse vector \mathbf{x} stored in the compressed form with a sparse uncompressed vector \mathbf{y} follows the algorithm

$$a = 0; \quad \text{for } j = 1, \dots, n_0 : \quad a = a + y_{\{i_j\}} \cdot c_j.$$

The compressed row storage for matrices is a generalization of the vector concept. We store the nonzero elements of A as a set of sparse row (or column) vectors in the compressed form. The main difference is that, instead of a single number n_0 , we need to store a whole vector \mathbf{n}_0 specifying the positions of the first row elements of A in \mathbf{c} . For example, the matrix

$$\mathbf{A} = \begin{pmatrix} A_{11} & A_{12} & 0 & 0 & 0 & 0 \\ A_{21} & 0 & 0 & A_{24} & 0 & 0 \\ 0 & 0 & 0 & A_{34} & 0 & 0 \\ 0 & 0 & A_{43} & 0 & A_{45} & 0 \\ 0 & A_{52} & 0 & 0 & 0 & A_{56} \end{pmatrix}$$

would be represented rowwise as

$$\begin{aligned} \mathbf{c} &= (A_{11}, A_{12} | A_{21}, A_{24} | A_{34} | A_{43}, A_{45} | A_{52}, A_{56}) , \\ \mathbf{i} &= (1, 2 | 1, 4 | 4 | 3, 5 | 2, 6) , \\ \mathbf{n}_0 &= (1, 3, 5, 6, 8, 10) . \end{aligned}$$

(The sign “|” just emphasizes the end of a row and has no consequence for the storage itself.) As in the case of vectors, the elements in each row do not have to be ordered. Consequently, there is no direct access to a particular element A_{ij} stored in \mathbf{c} . Nevertheless, retrieving a row is easy: it suffices to examine the part of \mathbf{i} corresponding to the i th row, which is given by \mathbf{n}_0 . On the contrary, retrieving a column involves a search through the whole storage scheme. Therefore, if a fast access to columns is necessary, it is preferable to simultaneously store A rowwise and columnwise.

A special type of sparse matrices are matrices with a banded structure.

Definition 2 The row bandwidth of a matrix $A \in \mathbb{R}^{m \times n}$ is defined as

2

$$w(A) = \max_{1 \leq i \leq m} (l_i(A) - f_i(A) + 1) ,$$

where $f_i(A) = \min\{j | A_{ij} \neq 0\}$ and $l_i(A) = \max\{j | A_{ij} \neq 0\}$ are column indices of the first and last nonzero elements in the i th row of A .

A banded matrix A is considered to be sparse if $w(A) \ll n$. Contrary to the general case, vector \mathbf{c} of a banded matrix typically contains for each row all elements between the first and last nonzero ones. Thus, the storage scheme does not have to include in \mathbf{i} all column indices, only one index for the first nonzero element in a row. On the other hand, zeros within the band have to be stored as well. For example, the matrix

$$A = \begin{pmatrix} A_{11} & A_{12} & 0 & 0 & 0 \\ 0 & A_{22} & 0 & A_{24} & 0 \\ 0 & 0 & 0 & A_{34} & 0 \\ 0 & 0 & A_{43} & 0 & A_{45} \end{pmatrix}$$

would be represented as

$$\begin{aligned} \mathbf{c} &= (A_{11}, A_{12} | A_{22}, 0, A_{24} | A_{34} | A_{43}, 0, A_{45}) , \\ \mathbf{i} &= (1, 2, 4, 3) , \\ \mathbf{n}_0 &= (1, 3, 6, 7, 10) . \end{aligned}$$

An interesting observation is that the row bandwidth $w(A)$ can be influenced by column permutations. The fill-minimizing column orderings are discussed by Björck (1996) and George and Ng (1983), for instance.

Details on some other storage schemes can be found in Duff et al. (1989) and Press et al. (1992).

4.5.2 Methods for Sparse Matrices

Methods for sparse matrices are still subject to intensive research. Moreover, the choice of a suitable method for a given problem (and even the choice of an algorithm for elementary operations such as matrix-vector multiplication) depends on many factors, including dimension, matrix type storage scheme, and computational environment (e.g., storage in virtual memory vs. auxiliary storage; vector vs. parallel computing, etc.). Therefore, we provide only a general overview and references to most general results. More details can be found in Björck (1996), Dongarra and Eijkhout (2000), Duff et al. (1989), Hackbusch (1994) and Saad (2003).

First, many discussed algorithms can be relatively easily adopted for banded matrices. For example, having a row-based storage scheme, one just needs to modify the summation limits in the row version of Cholesky decomposition. Moreover, the positions of nonzero elements can be determined in advance (Ng and Peyton, 1993).

Second, the algorithms for general sparse matrices are more complicated. A graph representation may be used to capture the nonzero pattern of a matrix as well as to predict the pattern of the result (e.g., the nonzero pattern of $A^T A$, the Cholesky factor U , etc.). To give an overview, methods adopted for sparse matrices include, but are not limited to, usually used decompositions (e.g., Cholesky, Ng and Peyton, 1993; LU and LDU, Mittal and Al-Kurdi, 2002; QR, George and Liu, 1987, and Heath, 1984), solving systems of equations by direct (Gupta, 2002; Tran et al., 1996) and iterative methods (Makinson and Shah, 1986; Zlatev and Nielsen, 1988) and searching eigenvalues (Bergamaschi and Putti, 2002; Golub et al., 2000).

References

- Alanelli, M. and Hadjidimos, A. (2004). Block Gauss elimination followed by a classical iterative method for the solution of linear systems. *Journal of Computational and Applied Mathematics*, 163(2): 381–400
- Anderson, E., Bai, Z., Bischof, C., Blackford, S., Demmel, J., Dongarra, J., Croz, J.D., Greenbaum, A., Hammarling, S., McKenney, A. and Sorensen, D. (1999). *LAPACK Users' Guide, Third Edition*. SIAM Press, Philadelphia, USA.
- Axelsson, O. (1994). *Iterative Solution Methods*. Cambridge University Press, Cambridge, UK.
- Bergamaschi, L. and Putti, M. (2002). Numerical comparison of iterative eigen-solvers for large sparse symmetric positive definite matrices. *Computer Methods in Applied Mechanics and Engineering*, 191: 5233–5247.
- Benoit, C. (1924). Note sur une méthode de résolution des équations normales provenant de l'application de la méthode des moindres carrés à un système

- d'équations linéaires en nombre inférieur à celui des inconnues. Application de la méthode à la résolution d'un système défini d'équations linéaires (Procédé du Commandant Cholesky). *Bulletin géodésique*, 2: 5–77.
- Björck, A. (1994). Numerics of Gram–Schmidt Orthogonalization. *Linear Algebra and its Applications*, 198: 297–316.
- Björck, A. (1996). *Numerical Methods for Least Squares Problems*. SIAM Press, Philadelphia, USA.
- Croz, J.D. and Higham, N.J. (1992). Stability of methods for matrix inversion. *IMA Journal of Numerical Analysis*, 12: 1–19.
- Dax, A. (2000). A modified Gram–Schmidt algorithm with iterative orthogonalization and column pivoting. *Linear Algebra and Its Applications*, 310: 25–42.
- Demmel, J.W., Gu, M., Eisenstat, S., Slapničar, I., Veselić, K. and Drmač, Z. (1999). Computing the singular value decomposition with high relative accuracy. *Linear Algebra and its Applications*, 299: 21–80.
- Dongarra, J.J. and Eijkhout, V. (2000). Numerical linear algebra algorithms and software. *Journal of Computational and Applied Mathematics*, 123: 489–514.
- Duff, I.S., Erisman, A.M. and Reid, J.K. (1989). *Direct Methods for Sparse Matrices*. Oxford University Press, USA.
- Freund, R. and Nachtigal, N. (1991). QMR: a quasi-minimal residual method for non-Hermitian linear systems. *Numerical Mathematics*, 60: 315–339.
- Gallivan, K.A., Plemmons, R.J. and Sameh, A.H. (1990). Parallel algorithms for dense linear algebra computations. *SIAM Review*, 32: 54–135.
- Gentle, J.E. (1998). *Numerical Linear Algebra for Applications in Statistics*. Springer, New York, USA.
- Gentleman, W.M. (1973). Least squares computations by Givens transformations without square roots. *Journal of Institute of Mathematics and its Applications*, 12: 329–336.
- Gentleman, W.M. (1975). Error analysis of QR decomposition by Givens transformations. *Linear Algebra and its Applications*, 10: 189–197.
- George, A. and Liu, J.W.H. (1987). Householder reflections versus givens rotations in sparse orthogonal decomposition. *Linear Algebra and its Applications*, 88: 223–238.
- George, J.A. and Ng, E.G. (1983). On row and column orderings for sparse least squares problems. *SIAM Journal of Numerical Analysis*, 20: 326–344.
- Givens, W. (1958). Computation of Plane Unitary Rotations Transforming a General Matrix to Triangular Form. *Journal of SIAM*, 6(1): 26–50.
- Golub, G.H. (1965). Numerical methods for solving least squares problems. *Numerical Mathematics*, 7: 206–216.
- Golub, G.H. and Kahan, W. (1965). Calculating the singular values and pseudo-inverse of a matrix. *SIAM Journal on Numerical Analysis B*, 2: 205–224.
- Golub, G.H. and Reinsch, C. (1970). Singular value decomposition and least squares solution. *Numerical Mathematics*, 14: 403–420.
- Golub, G.H. and van Loan, C.F. (1996). *Matrix Computations*. Johns Hopkins University Press, Baltimore, Maryland.

- Golub, G.H., Zhang, Z. and Zha, H. (2000). Large sparse symmetric eigenvalue problems with homogeneous linear constraints: the Lanczos process with inner-outer iterations. *Linear Algebra and its Applications*, 309: 289–306.
- Gupta, A. (2002). Recent Advances in Direct Methods for Solving Unsymmetric Sparse Systems of Linear Equations. *ACM Transactions on Mathematical Software*, 28: 301–324.
- Hackbusch, W. (1994). *Iterative Solution of Large Sparse Systems of Equations*. Springer, New York, USA.
- Hadjidimos, A. (2000). Successive Overrelaxation (SOR) and related methods. *Journal of Computational and Applied Mathematics*, 123: 177–199.
- Hammarling, S. (1974). A note on modifications to the Givens plane rotation. *Journal of Institute of Mathematics and its Applications*, 13: 215–218.
- Hari, V. and Veselić, K. (1987). On Jacobi methods for singular value decompositions. *SIAM Journal of Scientific and Statistical Computing*, 8: 741–754.
- Harville, D.A. (1997). *Matrix Algebra from a Statistician's Perspective*. Springer, New York, USA.
- Heath, M.T. (1984). Numerical methods for large sparse linear least squares problems. *SIAM Journal of Scientific and Statistical Computing*, 26: 497–513.
- Hestenes, M.R. and Stiefel, E. (1952). Method of conjugate gradients for solving linear systems. *J. Res. Nat Bur. Standards B*, 49: 409–436.
- Higham, N.J. (1989). The accuracy of solutions to triangular systems. *SIAM Journal on Numerical Analysis*, 26: 1252–1265.
- Higham, N.J. (1997). Recent Developments in Dense Numerical Linear Algebra. In Duff, I.S. and Watson, G.A. (eds), *State of the Art in Numerical Analysis*, Oxford University Press, Oxford.
- Higham, N.J. (2000). QR factorization with complete pivoting and accurate computation of the SVD. *Linear Algebra and its Applications*, 309: 153–174.
- Higham, N.J. (2002). *Accuracy and Stability of Numerical Algorithms*, Second edition. SIAM Press, Philadelphia, USA.
- Hong, Y.P. and Tan, C.T. (1992). Rank-revealing QR factorizations and the singular value decomposition. *Mathematics of Computation*, 58: 213–232.
- Householder, A.S. (1958). Unitary triangularization of a nonsymmetric matrix. *Journal of the Association of Computing Machinery*, 5: 339–342.
- Ipsen, I.C.F. (1997). Computing an Eigenvector with Inverse Iteration. *SIAM Review*, 39: 254–291.
- Kahan, W. (1958). *Gauss–Seidel methods of solving large systems of linear equations*. Doctoral thesis, University of Toronto, Toronto, Canada.
- Kammerer, W.J. and Nashed, M.Z. (1972). On the convergence of the conjugate gradient method for singular linear operator equations. *SIAM Journal on Numerical Analysis*, 9: 165–181.
- Makinson, G.J. and Shah, A.A. (1986). An iterative solution method for solving sparse nonsymmetric linear systems. *Journal of Computational and Applied Mathematics*, 15: 339–352.
- Martin, R.S., Peters, G. and Wilkinson, J.H. (1965). Symmetric decomposition of a positive definite matrix. In Wilkinson, J.H. and Reinsch, C. (eds), *Linear Al-*

- gebra (*Handbook for Automation Computation, Vol. II*). Springer, Heidelberg, Germany.
- Meinguet, J. (1983). Refined error analysis of cholesky factorization. *SIAM Journal on Numerical Analysis*, 20: 1243–1250.
- Milaszewicz, J.P. (1987). Improving Jacobi and Gauss–Seidel Iterations. *Linear Algebra and Its Applications*, 93: 161–170.
- Miranian, L. and Gu, M. (2003). Strong rank revealing LU factorizations. *Linear Algebra and its Applications*, 367: 1–16.
- Mittal, R.C. and Al-Kurdi, A. (2002). LU-decomposition and numerical structure for solving large sparse nonsymmetric linear systems. *Computers & Mathematics with Applications*, 43: 131–155.
- Ng, E.G. and Peyton, B.W. (1993). Block Sparse Cholesky Algorithm on Advanced Uniprocessor Computers. *SIAM Journal of Scientific Computing*, 14: 1034–1056.
- Nool, M. (1995). Explicit parallel block Cholesky algorithms on the CRAY APP. *Applied Numerical Mathematics*, 19: 91–114.
- Pan, C.T. (2000). On the existence and computation of rank revealing LU factorizations. *Linear Algebra and its Applications*, 316: 199–222.
- Press, W.H., Teukolsky, S.A., Vetterling, W.T. and Flannery, B.P. (1992). *Numerical Recipes in C: the Art of Scientific Computing, Second Edition*. Cambridge University Press, Cambridge, UK.
- Rice, J.R. (1966). Experiments on Gram–Schmidt orthogonalization. *Mathematics of Computation*, 20: 325–328.
- Saad, Y. (2003). *Iterative Methods for Sparse Linear Systems. Second Edition*. SIAM Press, USA.
- Saad, Y. and Schultz, M.H. (1986). GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM Journal of Scientific and Statistical Computing*, 7: 856–869.
- Sidi, A. (1989). On extensions of the power method for normal operators. *Linear Algebra and Its Applications*, 120: 207–224.
- Skeel, R.D. (1980). Iterative refinement implies numerical stability for Gaussian elimination. *Mathematics of Computation*, 35: 817–832.
- Stewart, G.W. (1976). The economical storage of plane rotations. *Numerical Mathematics*, 25: 137–138.
- Stewart, G.W. (1998). *Matrix Algorithms, Volume I: Basic Decompositions*. SIAM Press, Philadelphia, USA.
- Stoer, J. and Bulirsch, R. (2002). *Introduction to Numerical Analysis, Third Edition*. Springer, New York, USA.
- Tran, T.M., Gruber, R., Appert, K. and Wuthrich, S. (1996). A direct parallel sparse matrix solver. *Computer Physics Communications*, 96: 118–128.
- Trefethen, L.N. and Bau, D. (1997). *Numerical Linear Algebra*. SIAM Press, Philadelphia, USA.
- von Matt, U. (1995). The Orthogonal QD-Algorithm. In Moonen, M. and De Moor, B. (eds), *SVD and Signal Processing, III: Algorithms, Architectures and Applications*, Elsevier, Amsterdam.

- Vorst, V.D. (1992). Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems. *SIAM Journal of Scientific and Statistical Computing*, 13: 631–644.
- Wampler, R.H. (1970). A report on the accuracy of some widely used least squares computer programs. *Journal of American Statistical Association*, 65: 549–565.
- Young, D.M. (1954). Iterative methods for solving partial differential equations of elliptic type. *Transactions of the American Mathematical Society*, 76: 92–111.
- Zhou, B.B. and Brent, R.P. (2003). An efficient method for computing eigenvalues of a real normal matrix. *Journal of Parallel and Distributed Computing*, 63: 638–648.
- Zlatev, Z. and Nielsen, H.B. (1988). Solving large and sparse linear least-squares problems by conjugate gradient algorithms. *Computers & Mathematics with Applications*, 15: 185–202.
- Zou, Q. (1991). An observation on Gauss elimination. *Computers and Mathematical Applications*, 22: 69–70.

The EM Algorithm

II.5

Shu Kay Ng, Thriyambakam Krishnan, Geoffrey J. McLachlan

5.1	<i>Introduction</i>	138
	Maximum Likelihood Estimation.....	138
	EM Algorithm: Incomplete-Data Structure	139
	Overview of the Chapter	140
5.2	<i>Basic Theory of the EM Algorithm</i>	140
	The E- and M-Steps	140
	Generalized EM Algorithm	141
	Convergence of the EM Algorithm	142
	Rate of Convergence of the EM Algorithm	144
	Properties of the EM Algorithm.....	145
5.3	<i>Examples of the EM Algorithm</i>	145
	Example 1: Normal Mixtures	145
	Example 2: Censored Failure-Time Data	147
	Example 3: Nonapplicability of EM Algorithm	148
	Starting Values for EM Algorithm	150
	Provision of Standard Errors	151
5.4	<i>Variations on the EM Algorithm</i>	154
	Complicated E-Step	154
	Complicated M-Step	156
	Speeding up Convergence	159
5.5	<i>Miscellaneous Topics on the EM Algorithm</i>	162
	EM Algorithm for MAP Estimation	162
	Iterative Simulation Algorithms	163
	Further Applications of the EM Algorithm.....	164

5.1

Introduction

5.1.1

Maximum Likelihood Estimation

The Expectation-Maximization (EM) algorithm is a broadly applicable approach to the iterative computation of maximum likelihood (ML) estimates, useful in a variety of incomplete-data problems. Maximum likelihood estimation and likelihood-based inference are of central importance in statistical theory and data analysis. Maximum likelihood estimation is a general-purpose method with attractive properties. It is the most-often used estimation technique in the frequentist framework; it is also relevant in the Bayesian framework (Chap. III.11). Often Bayesian solutions are justified with the help of likelihoods and maximum likelihood estimates (MLE), and Bayesian solutions are similar to penalized likelihood estimates. Maximum likelihood estimation is an ubiquitous technique and is used extensively in every area where statistical techniques are used.

We assume that the observed data \mathbf{y} has probability density function (p.d.f.) $g(\mathbf{y}; \boldsymbol{\Psi})$, where $\boldsymbol{\Psi}$ is the vector containing the unknown parameters in the postulated form for the p.d.f. of Y . Our objective is to maximize the likelihood $L(\boldsymbol{\Psi}) = g(\mathbf{y}; \boldsymbol{\Psi})$ as a function of $\boldsymbol{\Psi}$, over the parameter space $\boldsymbol{\Omega}$. That is,

$$\partial L(\boldsymbol{\Psi})/\partial \boldsymbol{\Psi} = \mathbf{0},$$

or equivalently, on the log likelihood,

$$\partial \log L(\boldsymbol{\Psi})/\partial \boldsymbol{\Psi} = \mathbf{0}. \quad (5.1)$$

The aim of ML estimation is to determine an estimate $\hat{\boldsymbol{\Psi}}$, so that it defines a sequence of roots of (5.1) that is consistent and asymptotically efficient. Such a sequence is known to exist under suitable regularity conditions (Cramér, 1946). With probability tending to one, these roots correspond to local maxima in the interior of $\boldsymbol{\Omega}$. For estimation models in general, the likelihood usually has a global maximum in the interior of $\boldsymbol{\Omega}$. Then typically a sequence of roots of (5.1) with the desired asymptotic properties is provided by taking $\hat{\boldsymbol{\Psi}}$ to be the root that globally maximizes $L(\boldsymbol{\Psi})$; in this case, $\hat{\boldsymbol{\Psi}}$ is the MLE. We shall henceforth refer to $\hat{\boldsymbol{\Psi}}$ as the MLE, even in situations where it may not globally maximize the likelihood. Indeed, in some of the examples on mixture models (McLachlan and Peel, 2000, Chap. 3), the likelihood is unbounded. However, for these models there may still exist under the usual regularity conditions a sequence of roots of (5.1) with the properties of consistency, efficiency, and asymptotic normality (McLachlan and Basford, 1988, Chap. 12).

When the likelihood or log likelihood is quadratic in the parameters as in the case of independent normally distributed observations, its maximum can be obtained by solving a system of linear equations in parameters. However, often in practice the likelihood function is not quadratic giving rise to nonlinearity problems in ML estimation. Examples of such situations are: (a) models leading to

means which are nonlinear in parameters; (b) despite a possible linear structure, the likelihood is not quadratic in parameters due to, for instance, non-normal errors, missing data, or dependence.

Traditionally ML estimation in these situations has been carried out using numerical iterative methods of solution of equations such as the Newton–Raphson (NR) method and its variants like Fisher’s method of scoring. Under reasonable assumptions on $L(\Psi)$ and a sufficiently accurate starting value, the sequence of iterates $\{\Psi^{(k)}\}$ produced by the NR method enjoys local quadratic convergence to a solution Ψ^* of (5.1). Quadratic convergence is regarded as the major strength of the NR method. But in applications, these methods could be tedious analytically and computationally even in fairly simple cases; see McLachlan and Krishnan (1997, Sect. 1.3) and Meng and van Dyk (1997). The EM algorithm offers an attractive alternative in a variety of settings. It is now a popular tool for iterative ML estimation in a variety of problems involving missing data or incomplete information.

EM Algorithm: Incomplete-Data Structure

5.1.2

In the application of statistical methods, one is often faced with the problem of estimation of parameters when the likelihood function is complicated in structure resulting in difficult-to-compute maximization problems. This difficulty could be analytical or computational or both. Some examples are grouped, censored or truncated data, multivariate data with some missing observations, multiway frequency data with a complex cell probability structure, and data from mixtures of distributions. In many of these problems, it is often possible to formulate an associated statistical problem with the same parameters with “augmented data” from which it is possible to work out the MLE in an analytically and computationally simpler manner. The augmented data could be called the “complete data” and the available data could be called the “incomplete data”, and the corresponding likelihoods, the “complete-data likelihood” and the “incomplete-data likelihood”, respectively, and the corresponding ML estimations, the “complete-data problem” and the “incomplete-data problem”. The EM Algorithm is a generic method for computing the MLE of an incomplete-data problem by formulating an associated complete-data problem, and exploiting the simplicity of the MLE of the latter to compute the MLE of the former. The augmented part of the data could also be called “missing data”, with respect to the actual incomplete-data problem on hand. The missing data need not necessarily be missing in the practical sense of the word. It may just be a conceptually convenient technical device. Thus the phrase “incomplete data” is used quite broadly to represent a variety of statistical data models, including mixtures, convolutions, random effects, grouping, censoring, truncated and missing observations.

The EM algorithm is an iterative algorithm, in each iteration of which there are two steps, the Expectation Step (E-step) and the Maximization Step (M-step). A brief history of the EM algorithm can be found in McLachlan and Krishnan (1997, Sect. 1.8). The name EM algorithm was coined by Dempster et al. (1977), who synthesized earlier formulations of this algorithm in many particular cases and presented

a general formulation of this method of finding MLE in a variety of problems and provided an initial catalogue of problems where this method could be profitably applied. Since then the EM algorithm has been applied in a staggering variety of general statistical problems such as resolution of mixtures, multiway contingency tables, variance components estimation, factor analysis, as well as in specialized applications in such areas as genetics, medical imaging, and neural networks.

5.1.3 Overview of the Chapter

In Sect. 5.2, the basic theory of the EM algorithm is presented. In particular, the monotonicity of the algorithm, convergence, and rate of convergence properties are systematically examined. In Sect. 5.3, the EM methodology presented in this chapter is illustrated in some commonly occurring situations such as the fitting of normal mixtures and missing observations in terms of censored failure times. We also provide an example in which the EM algorithm may not be applicable. Consideration is given also to the two important issues associated with the use of the EM algorithm, namely the initialization of the EM and the provision of standard errors.

We discuss further modifications and extensions to the EM algorithm in Sect. 5.4. In particular, the extensions of the EM algorithm known as the Monte Carlo EM, ECM, ECME, AECM, and PX-EM algorithms are considered. With the considerable attention being given to the analysis of large data sets, as in typical data mining applications, recent work on speeding up the implementation of the EM algorithm is discussed. These include the IEM, SPIEM, the scalable EM algorithms, and the use of multiresolution kd-trees.

In Sect. 5.5, the relationship of the EM algorithm to other data augmentation techniques, such as the Gibbs sampler and MCMC methods is presented briefly. The Bayesian perspective is also included by showing how the EM algorithm and its variants can be adapted to compute the maximum *a posteriori* (MAP) estimate. We conclude the chapter with a brief account of the applications of the EM algorithm in such topical and interesting areas as Bioinformatics and Image Analysis.

5.2 Basic Theory of the EM Algorithm

5.2.1 The E- and M-Steps

Within the incomplete-data framework of the EM algorithm, we let x denote the vector containing the complete data and we let z denote the vector containing the missing data. Even when a problem does not at first appear to be an incomplete-data one, computation of the MLE is often greatly facilitated by artificially formulating it to be as such. This is because the EM algorithm exploits the reduced complexity of ML estimation given the complete data. For many statistical problems the complete-data likelihood has a nice form.

We let $g_c(\mathbf{x}; \Psi)$ denote the p.d.f. of the random vector \mathbf{X} corresponding to the complete-data vector \mathbf{x} . Then the complete-data log likelihood function that could be formed for Ψ if \mathbf{x} were fully observable is given by

$$\log L_c(\Psi) = \log g_c(\mathbf{x}; \Psi).$$

The EM algorithm approaches the problem of solving the incomplete-data likelihood (5.1) indirectly by proceeding iteratively in terms of $\log L_c(\Psi)$. As it is unobservable, it is replaced by its conditional expectation given \mathbf{y} , using the current fit for Ψ . On the $(k+1)$ th iteration of the EM algorithm,
E-Step: Compute $Q(\Psi; \Psi^{(k)})$, where

$$Q(\Psi; \Psi^{(k)}) = E_{\Psi^{(k)}}\{\log L_c(\Psi)|\mathbf{y}\}. \quad (5.2)$$

M-Step: Choose $\Psi^{(k+1)}$ to be any value of $\Psi \in \Omega$ that maximizes $Q(\Psi; \Psi^{(k)})$:

$$Q(\Psi^{(k+1)}; \Psi^{(k)}) \geq Q(\Psi; \Psi^{(k)}) \quad \forall \Psi \in \Omega. \quad (5.3)$$

In (5.2) and elsewhere in this chapter, the operator $E_{\Psi^{(k)}}$ denotes expectation using the parameter vector $\Psi^{(k)}$. The E- and M-steps are alternated repeatedly until convergence, which may be determined, for instance, by using a suitable stopping rule like $\|\Psi^{(k+1)} - \Psi^{(k)}\| < \varepsilon$ for some $\varepsilon > 0$ with some appropriate norm $\|\cdot\|$ or the difference $L(\Psi^{(k+1)}) - L(\Psi^{(k)})$ changes by an arbitrarily small amount in the case of convergence of the sequence of likelihood values $\{L(\Psi^{(k)})\}$.

It can be shown that both the E- and M-steps will have particularly simple forms when $g_c(\mathbf{x}; \Psi)$ is from an exponential family:

$$g_c(\mathbf{x}; \Psi) = b(\mathbf{x}) \exp\{c^\top(\Psi)\mathbf{t}(\mathbf{x})\} / a(\Psi), \quad (5.4)$$

where $\mathbf{t}(\mathbf{x})$ is a $k \times 1$ ($k \geq d$) vector of complete-data sufficient statistics and $c(\Psi)$ is a $k \times 1$ vector function of the $d \times 1$ parameter vector Ψ , and $a(\Psi)$ and $b(\mathbf{x})$ are scalar functions. Members of the exponential family include most common distributions, such as the multivariate normal, Poisson, multinomial and others. For exponential families, the E-step can be written as

$$Q(\Psi; \Psi^{(k)}) = E_{\Psi^{(k)}}(\log b(\mathbf{x})|\mathbf{y}) + c^\top(\Psi)\mathbf{t}^{(k)} - \log a(\Psi),$$

where $\mathbf{t}^{(k)} = E_{\Psi^{(k)}}\{\mathbf{t}(\mathbf{X})|\mathbf{y}\}$ is an estimator of the sufficient statistic. The M-step maximizes the Q-function with respect to Ψ ; but $E_{\Psi^{(k)}}(\log b(\mathbf{x})|\mathbf{y})$ does not depend on Ψ . Hence it is sufficient to write:

E-Step: Compute

$$\mathbf{t}^{(k)} = E_{\Psi^{(k)}}\{\mathbf{t}(\mathbf{X})|\mathbf{y}\}.$$

M-Step: Compute

$$\Psi^{(k+1)} = \arg \max_{\Psi} [c^\top(\Psi)\mathbf{t}^{(k)} - \log a(\Psi)].$$

In Example 2 of Sect. 5.3.2, the complete-data p.d.f. has an exponential family representation. We shall show how the implementation of the EM algorithm can be simplified.

5.2.2 Generalized EM Algorithm

Often in practice, the solution to the M-step exists in closed form. In those instances where it does not, it may not be feasible to attempt to find the value of Ψ that globally maximizes the function $Q(\Psi; \Psi^{(k)})$. For such situations, Dempster et al. (1977) defined a generalized EM (GEM) algorithm for which the M-Step requires $\Psi^{(k+1)}$ to be chosen such that

$$Q(\Psi^{(k+1)}; \Psi^{(k)}) \geq Q(\Psi^{(k)}; \Psi^{(k)}) \quad (5.5)$$

holds. That is, one chooses $\Psi^{(k+1)}$ to increase the Q -function, $Q(\Psi; \Psi^{(k)})$, over its value at $\Psi = \Psi^{(k)}$, rather than to maximize it over all $\Psi \in \Omega$ in (5.3).

It is of interest to note that the EM (GEM) algorithm as described above implicitly defines a mapping $\Psi \rightarrow M(\Psi)$, from the parameter space Ω to itself such that

$$\Psi^{(k+1)} = M(\Psi^{(k)}) \quad (k = 0, 1, 2, \dots).$$

The function M is called the EM mapping. We shall use this function in our subsequent discussion on the convergence property of the EM algorithm.

5.2.3 Convergence of the EM Algorithm

Let $k(x|y; \Psi) = g_c(x; \Psi)/g(y; \Psi)$ be the conditional density of X given $Y = y$. Then the complete-data log likelihood can be expressed by

$$\log L_c(\Psi) = \log g_c(x; \Psi) = \log L(\Psi) + \log k(x|y; \Psi). \quad (5.6)$$

Taking expectations on both sides of (5.6) with respect to the conditional distribution $x|y$ using the fit $\Psi^{(k)}$ for Ψ , we have

$$Q(\Psi; \Psi^{(k)}) = \log L(\Psi) + H(\Psi; \Psi^{(k)}), \quad (5.7)$$

where $H(\Psi; \Psi^{(k)}) = E_{\Psi^{(k)}}\{\log k(X|y; \Psi)|y\}$. It follows from (5.7) that

$$\begin{aligned} \log L(\Psi^{(k+1)}) - \log L(\Psi^{(k)}) &= \{Q(\Psi^{(k+1)}; \Psi^{(k)}) - Q(\Psi^{(k)}; \Psi^{(k)})\} \\ &\quad - \{H(\Psi^{(k+1)}; \Psi^{(k)}) - H(\Psi^{(k)}; \Psi^{(k)})\}. \end{aligned} \quad (5.8)$$

By Jensen's inequality, we have $H(\Psi^{(k+1)}; \Psi^{(k)}) \leq H(\Psi^{(k)}; \Psi^{(k)})$. From (5.3) or (5.5), the first difference on the right-hand side of (5.8) is nonnegative. Hence, the likelihood function is not decreased after an EM or GEM iteration:

$$L(\Psi^{(k+1)}) \geq L(\Psi^{(k)}) \quad (k = 0, 1, 2, \dots). \quad (5.9)$$

A consequence of (5.9) is the self-consistency of the EM algorithm. Thus for a bounded sequence of likelihood values $\{L(\Psi^{(k)})\}$, $L(\Psi^{(k)})$ converges monotonically to some L^* . Now questions naturally arise as to the conditions under which L^* corresponds to a stationary value and when this stationary value is at least a local maximum if not a global maximum. Examples are known where the EM algorithm converges to a local *minimum* and to a saddle point of the likelihood (McLachlan and Krishnan, 1997, Sect. 3.6). There are also questions of convergence of the sequence of EM iterates, that is, of the sequence of parameter values $\{\Psi^{(k)}\}$ to the MLE.

Before the general formulation of the EM algorithm in Dempster et al. (1977), there have been convergence results for special cases, notable among them being those of Baum et al. (1970) for what is now being called the hidden Markov model. In the article of Dempster et al. (1977) itself, there are some convergence results. However, it is Wu (1983) who investigates in detail several convergence issues of the EM algorithm in its generality. Wu examines these issues through their relationship to other optimization methods. He shows that when the complete data are from a curved exponential family with compact parameter space, and when the Q -function satisfies a certain mild differentiability condition, then any EM sequence converges to a stationary point (not necessarily a maximum) of the likelihood function. If $L(\Psi)$ has multiple stationary points, convergence of the EM sequence to either type (local or global maximizers, saddle points) depends upon the starting value $\Psi^{(0)}$ for Ψ . If $L(\Psi)$ is unimodal in Ω and satisfies the same differentiability condition, then any sequence $\{\Psi^{(k)}\}$ will converge to the unique MLE of Ψ , irrespective of its starting value.

To be more specific, one of the basic convergence results of the EM algorithm is the following:

$$\log L(M(\Psi)) \geq \log L(\Psi)$$

with equality if and only if

$$Q(M(\Psi); \Psi) = Q(\Psi; \Psi) \quad \text{and} \quad k(x|y; M(\Psi)) = k(x|y; \Psi).$$

This means that the likelihood function increases at each iteration of the EM algorithm, until the condition for equality is satisfied and a fixed point of the iteration is reached. If $\hat{\Psi}$ is an MLE, so that $\log L(\hat{\Psi}) \geq \log L(\Psi)$, $\forall \Psi \in \Omega$, then $\log L(M(\hat{\Psi})) = \log L(\hat{\Psi})$. Thus MLE are fixed points of the EM algorithm. If we have the likelihood function bounded (as might happen in many cases of interest), the EM sequence $\{\Psi^{(k)}\}$ yields a bounded nondecreasing sequence $\{\log L(\Psi^{(k)})\}$ which must converge as $k \rightarrow \infty$.

The theorem does not quite imply that fixed points of the EM algorithm are in fact MLEs. This is however true under fairly general conditions. For proofs and other details, see McLachlan and Krishnan (1997, Sect. 3.5) and Wu (1983). Furthermore, if a sequence of EM iterates $\{\Psi^{(k)}\}$ satisfy the conditions

1. $[\partial Q(\Psi; \Psi^{(k)})/\partial \Psi]_{\Psi=\Psi^{(k+1)}} = \mathbf{0}$, and
2. the sequence $\{\Psi^{(k)}\}$ converges to some value Ψ^* and $\log k(x|y; \Psi)$ is sufficiently smooth,

then we have $[\partial \log L(\Psi)/\partial \Psi]_{\Psi=\Psi^*} = \mathbf{0}$; see Little and Rubin (2002) and Wu (1983). Thus, despite the earlier convergence results, there is no guarantee that the convergence will be to a global maximum. For likelihood functions with multiple maxima, convergence will be to a local maximum which depends on the starting value $\Psi^{(0)}$.

Nettleton (1999) extends Wu's convergence results to the case of constrained parameter spaces and establishes some stricter conditions to guarantee convergence of the EM likelihood sequence to some local maximum and the EM parameter iterates to converge to the MLE.

5.2.4 Rate of Convergence of the EM Algorithm

The rate of convergence of the EM algorithm is also of practical interest. The convergence rate is usually slower than the quadratic convergence typically available with Newton-type methods. Dempster et al. (1977) show that the rate of convergence of the EM algorithm is linear and the rate depends on the proportion of information in the observed data. Thus in comparison to the formulated complete-data problem, if a large portion of data is missing, convergence can be quite slow.

Recall the EM mapping M defined in Sect. 5.2.2. If $\Psi^{(k)}$ converges to some point Ψ^* and $M(\Psi)$ is continuous, then Ψ^* is a fixed point of the algorithm; that is, Ψ^* must satisfy $\Psi^* = M(\Psi^*)$. By a Taylor series expansion of $\Psi^{(k+1)} = M(\Psi^{(k)})$ about the point $\Psi^{(k)} = \Psi^*$, we have in a neighborhood of Ψ^* that

$$\Psi^{(k+1)} - \Psi^* \approx J(\Psi^*)(\Psi^{(k)} - \Psi^*),$$

where $J(\Psi)$ is the $d \times d$ Jacobian matrix for $M(\Psi) = (M_1(\Psi), \dots, M_d(\Psi))^T$, having (i, j) th element $r_{ij}(\Psi)$ equal to

$$r_{ij}(\Psi) = \partial M_i(\Psi)/\partial \Psi_j,$$

where $\Psi_j = (\Psi)_j$ and d is the dimension of Ψ . Thus, in a neighborhood of Ψ^* , the EM algorithm is essentially a linear iteration with rate matrix $J(\Psi^*)$, since $J(\Psi^*)$ is typically nonzero. For this reason, $J(\Psi^*)$ is often referred to as the matrix rate of convergence. For vector Ψ , a measure of the actual observed convergence rate is the global rate of convergence, which is defined as

$$r = \lim_{k \rightarrow \infty} \frac{\|\Psi^{(k+1)} - \Psi^*\|}{\|\Psi^{(k)} - \Psi^*\|},$$

where $\|\cdot\|$ is any norm on d -dimensional Euclidean space \mathfrak{R}^d . It is noted that the observed rate of convergence equals the largest eigenvalue of $J(\Psi^*)$ under certain regularity conditions (Meng and van Dyk, 1997). As a large value of r implies slow convergence, the global speed of convergence is defined to be $s = 1 - r$ (Meng, 1994).

Properties of the EM Algorithm

5.2.5

The EM algorithm has several appealing properties, some of which are:

1. It is numerically stable with each EM iteration increasing the likelihood.
2. Under fairly general conditions, it has reliable global convergence.
3. It is easily implemented, analytically and computationally. In particular, it is generally easy to program and requires small storage space. By watching the monotone increase in likelihood (if evaluated easily) over iterations, it is easy to monitor convergence and programming errors (McLachlan and Krishnan, 1997, Sect. 1.7).
4. The cost per iteration is generally low, which can offset the larger number of iterations needed for the EM algorithm compared to other competing procedures.
5. It can be used to provide estimates of missing data.

Some of its drawbacks are:

1. It does not automatically provide an estimate of the covariance matrix of the parameter estimates. However, this disadvantage can be easily removed by using appropriate methodology associated with the EM algorithm (McLachlan and Krishnan, 1997, Chap. 4).
2. It is sometimes very slow to converge.
3. In some problems, the E- or M-steps may be analytically intractable.

We shall briefly address these three issues in Sects. 5.3.5 and 5.4.

Examples of the EM Algorithm

5.3

Example 1: Normal Mixtures

5.3.1

One of the classical formulations of the two-group discriminant analysis or the statistical pattern recognition problem involves a mixture of two p -dimensional normal distributions with a common covariance matrix. The problem of two-group cluster analysis with multiple continuous observations has also been formulated in this way. Here, we have n independent observations y_1, y_2, \dots, y_n from the mixture density

$$(1 - \pi)\phi(\mathbf{y}; \boldsymbol{\mu}_1, \boldsymbol{\Sigma}) + \pi\phi(\mathbf{y}; \boldsymbol{\mu}_2, \boldsymbol{\Sigma}),$$

where $\phi(\mathbf{y}; \boldsymbol{\mu}_i, \boldsymbol{\Sigma})$ denotes the p -dimensional normal density function with mean vector $\boldsymbol{\mu}_i$ and common covariance matrix $\boldsymbol{\Sigma}$, $i = 1, 2$. The $(1 - \pi)$ and π denote the proportions of the two clusters, respectively. The problem of estimating the parameters $\boldsymbol{\Psi} = (\pi, \boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \boldsymbol{\Sigma})$ is an instance of the problem of resolution of mixtures or in pattern recognition parlance an “unsupervised learning problem”. The

MLE problem here is quite messy and classical statistical and pattern recognition literature has struggled with it for a long time.

Consider the corresponding “supervised learning problem”, where observations on the random vector $\mathbf{X} = (Z, \mathbf{Y})$ are $\mathbf{x}_1 = (z_1, \mathbf{y}_1)$, $\mathbf{x}_2 = (z_2, \mathbf{y}_2)$, \dots , $\mathbf{x}_n = (z_n, \mathbf{y}_n)$. Here z_j is an indicator variable which identifies the j th observation as coming from the first ($z = 0$) or the second ($z = 1$) component ($j = 1, \dots, n$). The MLE problem is far simpler here with easy closed-form MLE. The classificatory variable z_j could be called the “missing variable” and data $\mathbf{z} = (z_1, z_2, \dots, z_n)^\top$ the missing data. The unsupervised learning problem could be called the incomplete-data problem and the supervised learning problem the complete-data problem. A relatively simple iterative method for computing the MLE for the unsupervised problem could be given exploiting the simplicity of the MLE for the supervised problem. This is the essence of the EM algorithm.

The complete-data log likelihood function for Ψ is given by

$$\log L_c(\Psi) = \sum_{j=1}^n (1 - z_j) \log \phi(\mathbf{y}_j; \boldsymbol{\mu}_1, \boldsymbol{\Sigma}) + z_j \log \phi(\mathbf{y}_j; \boldsymbol{\mu}_2, \boldsymbol{\Sigma}). \quad (5.10)$$

By differentiating (5.10) with respect to Ψ , the MLEs of Ψ are obtained, as if \mathbf{z} were actually observed:

$$\pi = \sum_{j=1}^n z_j / n, \quad (5.11)$$

$$\boldsymbol{\mu}_1 = \sum_{j=1}^n (1 - z_j) \mathbf{y}_j / \left(n - \sum_{j=1}^n z_j \right), \quad \boldsymbol{\mu}_2 = \sum_{j=1}^n z_j \mathbf{y}_j / \sum_{j=1}^n z_j, \quad (5.12)$$

$$\boldsymbol{\Sigma} = \sum_{j=1}^n \left[(1 - z_j) (\mathbf{y}_j - \boldsymbol{\mu}_1) (\mathbf{y}_j - \boldsymbol{\mu}_1)^\top + z_j (\mathbf{y}_j - \boldsymbol{\mu}_2) (\mathbf{y}_j - \boldsymbol{\mu}_2)^\top \right] / n, \quad (5.13)$$

Now the EM algorithm for this problem starts with some initial value $\Psi^{(0)}$ for the parameters. As $\log L_c(\Psi)$ in (5.10) is a linear function of the unobservable data \mathbf{z} for this problem, the calculation of $Q(\Psi; \Psi^{(k)})$ on the E-step is effected simply by replacing z_j by its current conditional expectation given the observed data \mathbf{y} , which is the usual posterior probability of the j th observation arising from component 2

$$\tau_j^{(k)} = E_{\Psi^{(k)}}(Z_j | \mathbf{y}) = \frac{\pi^{(k)} \phi(\mathbf{y}_j; \boldsymbol{\mu}_2^{(k)}, \boldsymbol{\Sigma}^{(k)})}{(1 - \pi^{(k)}) \phi(\mathbf{y}_j; \boldsymbol{\mu}_1^{(k)}, \boldsymbol{\Sigma}^{(k)}) + \pi^{(k)} \phi(\mathbf{y}_j; \boldsymbol{\mu}_2^{(k)}, \boldsymbol{\Sigma}^{(k)})}.$$

The M-step then consists of substituting these $\tau_j^{(k)}$ values for z_j in (5.11) to (5.13). The E- and M-steps are then iterated until convergence. Unlike in the MLE for the supervised problem, in the M-step of the unsupervised problem, the posterior probabilities τ_j , which are between 0 and 1, are used. The mean vectors $\boldsymbol{\mu}_i$ ($i = 1, 2$) and the covariance matrix $\boldsymbol{\Sigma}$ are computed using the $\tau_j^{(k)}$ as weights in weighted averages.

It is easy to extend the above method to a mixture of $g > 2$ multinormal mixtures or even to a mixture of $g > 2$ distributions from other (identifiable) families. For a detailed discussion of the applications of the EM algorithm in the resolution of finite mixtures and other issues of finite mixtures, see McLachlan and Peel (2000).

Example 2: Censored Failure-Time Data

5.3.2

In survival or reliability analyses, the focus is the distribution of time T to the occurrence of some event that represents failure (for computational methods in survival analysis see also Chap. III.12). In many situations, there will be individuals who do not fail at the end of the study, or individuals who withdraw from the study before it ends. Such observations are censored, as we know only that their failure times are greater than particular values. We let $\mathbf{y} = (c_1, \delta_1, \dots, c_n, \delta_n)^\top$ denote the observed failure-time data, where $\delta_j = 0$ or 1 according as the j th observation T_j is censored or uncensored at c_j ($j = 1, \dots, n$). That is, if T_j is uncensored, $t_j = c_j$, whereas if $t_j > c_j$, it is censored at c_j .

In the particular case where the p.d.f. for T is exponential with mean μ , we have

$$f(t; \mu) = \mu^{-1} \exp(-t/\mu) I_{(0, \infty)}(t) \quad (\mu > 0), \quad (5.14)$$

where the indicator function $I_{(0, \infty)}(t) = 1$ for $t > 0$ and is zero elsewhere. The unknown parameter vector Ψ is now a scalar, being equal to μ . Denote by s the number of uncensored observations. By re-ordering the data so that the uncensored observations precede censored observations. It can be shown that the log likelihood function for μ is given by

$$\log L(\mu) = -s \log \mu - \sum_{j=1}^n c_j / \mu. \quad (5.15)$$

By equating the derivative of (5.15) to zero, the MLE of μ is

$$\hat{\mu} = \sum_{j=1}^n c_j / s. \quad (5.16)$$

Thus there is no need for the iterative computation of $\hat{\mu}$. But in this simple case, it is instructive to demonstrate how the EM algorithm would work and how its implementation could be simplified as the complete-data log likelihood belongs to the regular exponential family (see Sect. 5.2.1).

The complete-data vector \mathbf{x} can be declared to be $\mathbf{x} = (t_1, \dots, t_s, \mathbf{z}^\top)^\top$, where $\mathbf{z} = (t_{s+1}, \dots, t_n)^\top$ contains the unobservable realizations of the $n - s$ censored random variables. The complete-data log likelihood is given by

$$\log L_c(\mu) = -n \log \mu - \sum_{j=1}^n t_j / \mu. \quad (5.17)$$

As $\log L_c(\mu)$ is a linear function of the unobservable data \mathbf{z} , the E-step is effected simply by replacing \mathbf{z} by its current conditional expectation given \mathbf{y} . By the lack of memory of the exponential distribution, the conditional distribution of $T_j - c_j$ given that $T_j > c_j$ is still exponential with mean μ . So, we have

$$E_{\mu^{(k)}}(T_j|\mathbf{y}) = E_{\mu^{(k)}}(T_j|T_j > c_j) = c_j + \mu^{(k)} \quad (5.18)$$

for $j = s + 1, \dots, n$. Accordingly, the Q-function is given by

$$Q(\mu; \mu^{(k)}) = -n \log \mu - \mu^{-1} \left\{ \sum_{j=1}^n c_j + (n-s)\mu^{(k)} \right\}.$$

In the M-step, we have

$$\mu^{(k+1)} = \left\{ \sum_{j=1}^n c_j + (n-s)\mu^{(k)} \right\} / n. \quad (5.19)$$

On putting $\mu^{(k+1)} = \mu^{(k)} = \mu^*$ in (5.19) and solving for μ^* , we have for $s < n$ that $\mu^* = \hat{\mu}$. That is, the EM sequence $\{\mu^{(k)}\}$ has the MLE $\hat{\mu}$ as its unique limit point, as $k \rightarrow \infty$; see McLachlan and Krishnan (1997, Sect. 1.5.2).

From (5.17), it can be seen that $\log L_c(\mu)$ has the exponential family form (5.4) with canonical parameter μ^{-1} and sufficient statistic $\mathbf{t}(X) = \sum_{j=1}^n T_j$. Hence, from (5.18), the E-step requires the calculation of $\mathbf{t}^{(k)} = \sum_{j=1}^n c_j + (n-s)\mu^{(k)}$. The M-step then yields $\mu^{(k+1)}$ as the value of μ that satisfies the equation

$$\mathbf{t}^{(k)} = E_{\mu}\{\mathbf{t}(X)\} = n\mu.$$

This latter equation can be seen to be equivalent to (5.19), as derived by direct differentiation of the Q-function.

5.3.3 Example 3: Nonapplicability of EM Algorithm

Examples 1 and 2 may have given an impression that the E-step consists in replacing the missing data by their conditional expectations given the observed data at current parameter values. Although in many examples this may be the case as $\log L_c(\Psi)$ is a linear function of the missing data \mathbf{z} , it is not quite so in general. Rather, as should be clear from the general theory described in Sect. 5.2.1, the E-step consists in replacing $\log L_c(\Psi)$ by its conditional expectation given the observed data at current parameter values. Flury and Zoppé (2000) give the following interesting example to demonstrate the point that the E-step does not always consist in plugging in “estimates” for missing data. This is also an example where the E-step cannot be correctly executed at all since the expected value of the complete-data log likelihood does not exist, showing thereby that the EM algorithm is not applicable to this problem, at least for this formulation of the complete-data problem.

Let the lifetimes of electric light bulbs of a certain type have a uniform distribution in the interval $(0, \theta]$, $\theta > 0$ and unknown. A total of $n + m$ bulbs are tested in two independent experiments. The observed data consist of $\mathbf{y} = (y_1, \dots, y_n)$ and $\mathbf{e} = (e_{n+1}, \dots, e_{n+m})$, where \mathbf{y} are exact lifetimes of a random sample of n bulbs and \mathbf{e} are indicator observations on a random sample of m bulbs, taking value 1 if the bulb is still burning at a fixed time point $T > 0$ and 0 if it is expired. The missing data is $\mathbf{z} = (y_{n+1}, \dots, y_{n+m})^\top$. Let s be the number of e_j 's with value 1 and $y_{\max} = \max\{y_1, \dots, y_n\}$.

In this example, the unknown parameter vector Ψ is a scalar, being equal to θ . Let us first work out the MLE of θ directly. The likelihood is

$$L(\theta) = \theta^{-n} I_{[y_{\max}, \infty)}(\theta) \times \left(\frac{T}{\max(T, \theta)} \right)^{m-s} \left(1 - \frac{T}{\max(T, \theta)} \right)^s,$$

where I_A is the notation for the indicator function of set A . For $s = 0$, $L(\theta)$ is decreasing in θ for $\theta \geq y_{\max}$ and hence the MLE is $\hat{\theta} = y_{\max}$. For $s \geq 1$, we have $\max(T, \theta) = \theta$. Here the function $L_1(\theta) = (\theta)^{-(n+m)} (\theta - T)^s$ has a unique maximum at $\tilde{\theta} = \frac{n+m}{n+m-s} T$ and is monotonically decreasing for $\theta > \tilde{\theta}$. Hence the MLE of θ is

$$\hat{\theta} = \begin{cases} \tilde{\theta} & \text{if } \tilde{\theta} > y_{\max} \text{ and } s \geq 1 \\ y_{\max} & \text{otherwise.} \end{cases}$$

Now let us try the EM algorithm for the case $s \geq 1$. The complete data can be formulated as $y_1, \dots, y_n, y_{n+1}, \dots, y_{n+m}$ and the complete-data MLE is

$$\max_{j=1, \dots, n+m} y_j.$$

Since $s \geq 1$, we have $\theta \geq T$. Now if we take the approach of replacing the missing observations, then we compute

$$E_{\theta^{(k)}} \left(y_j^{(k+1)} | \mathbf{y}, \mathbf{e} \right) = E_{\theta^{(k)}} (y_j | e_j) = \begin{cases} \frac{1}{2}(T + \theta) & \text{if } e_j = 1 \\ \frac{1}{2}T & \text{if } e_j = 0 \end{cases}$$

for $j = n + 1, \dots, n + m$. The M-step is

$$\theta^{(k+1)} = \max \left\{ y_{\max}, \frac{1}{2} (T + \theta^{(k)}) \right\}.$$

Combining the E- and M-steps, we can write the EM algorithm as a sequence of iterations of the equation

$$\theta^{(k+1)} = M(\theta^k) \equiv \max \left\{ y_{\max}, \frac{1}{2} (T + \theta^{(k)}) \right\}.$$

It is easily seen that if we start with any $\theta^{(0)}$, this procedure will converge to $\hat{\theta} = \max\{y_{\max}, T\}$, by noting that $\hat{\theta} = M(\hat{\theta})$.

The reason for the apparent EM algorithm not resulting in the MLE is that the E-step is wrong. In the E-step, we are supposed to find the conditional expectation of $\log L_c(\theta)$ given \mathbf{y} , \mathbf{e} at current parameter values. Now given the data with $s \geq 1$, we have $\theta \geq T$ and hence the conditional distributions of y_j are uniform in $[T, \theta^{(k)}]$. Thus for $\theta < \theta^{(k)}$ the conditional density of missing y_j takes value 0 with positive probability and hence the conditional expected value of the complete-data log likelihood we are seeking does not exist.

5.3.4 Starting Values for EM Algorithm

The EM algorithm will converge very slowly if a poor choice of initial value $\Psi^{(0)}$ were used. Indeed, in some cases where the likelihood is unbounded on the edge of the parameter space, the sequence of estimates $\{\Psi^{(k)}\}$ generated by the EM algorithm may diverge if $\Psi^{(0)}$ is chosen too close to the boundary. Also, with applications where the likelihood equation has multiple roots corresponding to local maxima, the EM algorithm should be applied from a wide choice of starting values in any search for all local maxima. A variation of the EM algorithm (Wright and Kennedy, 2000) uses interval analysis methods to locate multiple stationary points of a log likelihood within any designated region of the parameter space.

Here, we illustrate different ways of specification of initial value within mixture models framework. For independent data in the case of mixture models of g components, the effect of the E-step is to update the posterior probabilities of component membership. Hence the first E-step can be performed by specifying a value $\tau_j^{(0)}$ for each j ($j = 1, \dots, n$), where $\tau_j = (\tau_{1j}, \dots, \tau_{gj})^\top$ is the vector containing the g posterior probabilities of component membership for y_j . The latter is usually undertaken by setting $\tau_j^{(0)} = \mathbf{z}_j^{(0)}$, where

$$\mathbf{z}^{(0)} = \left(\mathbf{z}_1^{(0)\top}, \dots, \mathbf{z}_n^{(0)\top} \right)^\top$$

defines an initial partition of the data into g components. For example, an ad hoc way of initially partitioning the data in the case of, say, a mixture of $g = 2$ normal components with the same covariance matrices (Example 1, Sect. 5.3.1) would be to plot the data for selections of two of the p variables, and then draw a line that divides the bivariate data into two groups that have a scatter that appears normal. For higher-dimensional data, an initial value $\mathbf{z}^{(0)}$ for \mathbf{z} might be obtained through the use of some clustering algorithm, such as k -means or, say, an hierarchical procedure if n is not too large.

Another way of specifying an initial partition $\mathbf{z}^{(0)}$ of the data is to randomly divide the data into g groups corresponding to the g components of the mixture model. With random starts, the effect of the central limit theorem tends to have the component parameters initially being similar at least in large samples. One way to reduce this effect is to first select a small random subsample from the data, which is then randomly assigned to the g components. The first M-step is then performed on the basis of the subsample. The subsample has to be sufficiently

large to ensure that the first M-step is able to produce a nondegenerate estimate of the parameter vector Ψ (McLachlan and Peel, 2000, Sect. 2.12). In the context of g normal components, a method of specifying a random start is to generate the means $\mu_i^{(0)}$ independently as

$$\mu_1^{(0)}, \dots, \mu_g^{(0)} \stackrel{i.i.d.}{\sim} N(\bar{y}, V),$$

where \bar{y} is the sample mean and $V = \sum_{j=1}^n (y_j - \bar{y})(y_j - \bar{y})^\top / n$ is the sample covariance matrix of the observed data. With this method, there is more variation between the initial values $\mu_i^{(0)}$ for the component means μ_i than with a random partition of the data into g components. The component-covariance matrices Σ_i and the mixing proportions π_i can be specified as

$$\Sigma_i^{(0)} = V \text{ and } \pi_i^{(0)} = 1/g \quad (i = 1, \dots, g).$$

Ueda and Nakano (1998) considered a deterministic annealing EM (DAEM) algorithm in order for the EM iterative process to be able to recover from a poor choice of starting value. They proposed using the principle of maximum entropy and the statistical mechanics analogy, whereby a parameter, say θ , is introduced with $1/\theta$ corresponding to the “temperature” in an annealing sense. With their DAEM algorithm, the E-step is effected by averaging $\log L_c(\Psi)$ over the distribution taken to be proportional to that of the current estimate of the conditional density of the complete data (given the observed data) raised to the power of θ ; see for example McLachlan and Peel (2000, pp. 58–60).

Provision of Standard Errors

5.3.5

Several methods have been suggested in the EM literature for augmenting the EM computation with some computation for obtaining an estimate of the covariance matrix of the computed MLE. Many such methods attempt to exploit the computations in the EM steps. These methods are based on the observed information matrix $I(\hat{\Psi}; y)$, the expected information matrix $\mathcal{I}(\Psi)$ or on resampling methods. Baker (1992) reviews such methods and also develops a method for computing the observed information matrix in the case of categorical data. Jamshidian and Jennrich (2000) review more recent methods including the Supplemented EM (SEM) algorithm of Meng and Rubin (1991) and suggest some newer methods based on numerical differentiation.

Theoretically one may compute the asymptotic covariance matrix by inverting the observed or expected information matrix at the MLE. In practice, however, this may be tedious analytically or computationally, defeating one of the advantages of the EM approach. Louis (1982) extracts the observed information matrix in terms of the conditional moments of the gradient and curvature of the complete-data log likelihood function introduced within the EM framework. These conditional moments are generally easier to work out than the corresponding derivatives of the incomplete-data log likelihood function. An alternative approach is to numerically

differentiate the likelihood function to obtain the Hessian. In a EM-aided differentiation approach, Meilijson (1989) suggests perturbation of the incomplete-data score vector to compute the observed information matrix. In the SEM algorithm (Meng and Rubin, 1991), numerical techniques are used to compute the derivative of the EM operator M to obtain the observed information matrix. The basic idea is to use the fact that the rate of convergence is governed by the fraction of the missing information to find the increased variability due to missing information to add to the assessed complete-data covariance matrix. More specifically, let V denote the asymptotic covariance matrix of the MLE $\hat{\Psi}$. Meng and Rubin (1991) show that

$$I^{-1}(\hat{\Psi}; \mathbf{y}) = \mathcal{I}_c^{-1}(\hat{\Psi}; \mathbf{y}) + \Delta V, \quad (5.20)$$

where $\Delta V = \{I_d - J(\hat{\Psi})\}^{-1} J(\hat{\Psi}) \mathcal{I}_c^{-1}(\hat{\Psi}; \mathbf{y})$ and $\mathcal{I}_c(\hat{\Psi}; \mathbf{y})$ is the conditional expected complete-data information matrix, and where I_d denotes the $d \times d$ identity matrix. Thus the diagonal elements of ΔV give the increases in the asymptotic variances of the components of $\hat{\Psi}$ due to missing data. For a wide class of problems where the complete-data density is from the regular exponential family, the evaluation of $\mathcal{I}_c(\hat{\Psi}; \mathbf{y})$ is readily facilitated by standard complete-data computations (McLachlan and Krishnan, 1997, Sect. 4.5). The calculation of $J(\hat{\Psi})$ can be readily obtained by using only EM code via numerical differentiation of $M(\Psi)$. Let $\hat{\Psi} = \Psi^{(k+1)}$ where the sequence of EM iterates has been stopped according to a suitable stopping rule. Let M_i be the i th component of $M(\Psi)$. Let $\mathbf{u}^{(j)}$ be a column d -vector with the j th coordinate 1 and others 0. With a possibly different EM sequence $\Psi^{(k)}$, let r_{ij} be the (i, j) th element of $J(\hat{\Psi})$, we have

$$r_{ij}^{(k)} = \frac{M_i \left[\hat{\Psi} + \left(\Psi_j^{(k)} - \hat{\Psi}_j \mathbf{u}^{(j)} \right) \right] - \hat{\Psi}_i}{\Psi_j^{(k)} - \hat{\Psi}_j}.$$

Use a suitable stopping rule like $|r_{ij}^{(k+1)} - r_{ij}^{(k)}| < \sqrt{\varepsilon}$ to stop each of the sequences r_{ij} ($i, j = 1, 2, \dots, d$) and take $r_{ij}^* = r_{ij}^{(k+1)}$; see McLachlan and Krishnan (1997, Sect. 4.5).

It is important to emphasize that estimates of the covariance matrix of the MLE based on the expected or observed information matrices are guaranteed to be valid inferentially only asymptotically. In particular for mixture models, it is well known that the sample size n has to be very large before the asymptotic theory of maximum likelihood applies. A resampling approach, the bootstrap (Efron, 1979; Efron and Tibshirani, 1993), has been considered to tackle this problem. Basford et al. (1997) compared the bootstrap and information-based approaches for some normal mixture models and found that unless the sample size was very large, the standard errors obtained by an information-based approach were too unstable to be recommended.

The bootstrap is a powerful technique that permits the variability in a random quantity to be assessed using just the data at hand. Standard error estimation of $\hat{\Psi}$

may be implemented according to the bootstrap as follows. Further discussion on bootstrap and resampling methods can be found in Chaps. III.2 and III.3 of this handbook.

1. A new set of data, \mathbf{y}^* , called the bootstrap sample, is generated according to \hat{F} , an estimate of the distribution function of Y formed from the original observed data \mathbf{y} . That is, in the case where \mathbf{y} contains the observed values of a random sample of size n , \mathbf{y}^* consists of the observed values of the random sample

$$Y_1^*, \dots, Y_n^* \stackrel{\text{i.i.d.}}{\sim} \hat{F},$$

where the estimate \hat{F} (now denoting the distribution function of a single observation Y_j) is held fixed at its observed value.

2. The EM algorithm is applied to the bootstrap observed data \mathbf{y}^* to compute the MLE for this data set, $\hat{\Psi}^*$.
3. The bootstrap covariance matrix of $\hat{\Psi}^*$ is given by

$$\text{Cov}^*(\hat{\Psi}^*) = E^* \left[\{ \hat{\Psi}^* - E^*(\hat{\Psi}^*) \} \{ \hat{\Psi}^* - E^*(\hat{\Psi}^*) \}^\top \right], \quad (5.21)$$

where E^* denotes expectation over the bootstrap distribution specified by \hat{F} .

The bootstrap covariance matrix can be approximated by Monte Carlo methods. Steps 1 and 2 are repeated independently a number of times (say, B) to give B independent realizations of $\hat{\Psi}^*$, denoted by $\hat{\Psi}_1^*, \dots, \hat{\Psi}_B^*$. Then (5.21) can be approximated by the sample covariance matrix of these B bootstrap replications to give

$$\text{Cov}^*(\hat{\Psi}^*) \approx \sum_{b=1}^B (\hat{\Psi}_b^* - \bar{\hat{\Psi}}^*) (\hat{\Psi}_b^* - \bar{\hat{\Psi}}^*)^\top / (B-1), \quad (5.22)$$

where $\bar{\hat{\Psi}}^* = \sum_{b=1}^B \hat{\Psi}_b^* / B$. The standard error of the i th element of $\hat{\Psi}$ can be estimated by the positive square root of the i th diagonal element of (5.22). It has been shown that 50 to 100 bootstrap replications are generally sufficient for standard error estimation (Efron and Tibshirani, 1993).

In Step 1 above, the nonparametric version of the bootstrap would take \hat{F} to be the empirical distribution function formed from the observed data \mathbf{y} . Situations where we may wish to use the latter include problems where the observed data are censored or are missing in the conventional sense. In these cases the use of the nonparametric bootstrap avoids having to postulate a suitable model for the underlying mechanism that controls the censorship or the absence of the data. A generalization of the nonparametric version of the bootstrap, known as the weighted bootstrap, has been studied by Newton and Raftery (1994).

5.4

Variations on the EM Algorithm

In this section, further modifications and extensions to the EM algorithm are considered. In general, there are extensions of the EM algorithm

1. to produce standard errors of the MLE using the EM;
2. to surmount problems of difficult E-step and/or M-step computations;
3. to tackle problems of slow convergence;
4. in the direction of Bayesian or regularized or penalized ML estimations.

We have already discussed methods like the SEM algorithm for producing standard errors of EM-computed MLE in Sect. 5.3.5. The modification of the EM algorithm for Bayesian inference will be discussed in Sect. 5.5.1. In this section, we shall focus on the problems of complicated E- or M-steps and of slow convergence of the EM algorithm.

5.4.1

Complicated E-Step

In some applications of the EM algorithm, the E-step is complex and does not admit a close-form solution to the Q -function. In this case, the E-step at the $(k + 1)$ th iteration may be executed by a Monte Carlo (MC) process:

1. Make M independent draws of the missing values \mathbf{Z} , $\mathbf{z}^{(1_k)}, \dots, \mathbf{z}^{(M_k)}$, from the conditional distribution $k(\mathbf{z}|\mathbf{y}; \Psi^{(k)})$.
2. Approximate the Q -function as

$$Q(\Psi; \Psi^{(k)}) \approx Q_M(\Psi; \Psi^{(k)}) = \frac{1}{M} \sum_{m=1}^M \log k(\Psi|\mathbf{z}^{(m_k)}; \mathbf{y}).$$

In the M-step, the Q -function is maximized over Ψ to obtain $\Psi^{(k+1)}$. The variant is known as the Monte Carlo EM (MCEM) algorithm (Wei and Tanner, 1990). As MC error is introduced at the E-step, the monotonicity property is lost. But in certain cases, the algorithm gets close to a maximizer with a high probability (Booth and Hobert, 1999). The problems of specifying M and monitoring convergence are of central importance in the routine use of the algorithm. Wei and Tanner (1990) recommend small values of M be used in initial stages and be increased as the algorithm moves closer to convergence. As to monitoring convergence, they recommend that the values of $\Psi^{(k)}$ be plotted against k and when convergence is indicated by the stabilization of the process with random fluctuations about $\widehat{\Psi}$, the process may be terminated or continued with a larger value of M . Alternative schemes for specifying M and stopping rule are considered by Booth and Hobert (1999) and McCulloch (1997).

Example 4: Generalized Linear Mixed Models

Generalized linear mixed models (GLMM) are extensions of generalized linear models (GLM) (McCullagh and Nelder, 1989) that incorporate random effects in the linear predictor of the GLM (more material on the GLM can be found in

Chap. III.7). We let $\mathbf{y} = (y_1, \dots, y_n)^\top$ denote the observed data vector. Conditional on the unobservable random effects vector, $\mathbf{u} = (u_1, \dots, u_q)^\top$, we assume that \mathbf{y} arise from a GLM. The conditional mean $\mu_j = E(y_j|\mathbf{u})$ is related to the linear predictor $\eta_j = \mathbf{x}_j^\top \boldsymbol{\beta} + \mathbf{z}_j^\top \mathbf{u}$ by the link function $g(\mu_j) = \eta_j$ ($j = 1, \dots, n$), where $\boldsymbol{\beta}$ is a p -vector of fixed effects and \mathbf{x}_j and \mathbf{z}_j are, respectively, p -vector and q -vector of explanatory variables associated with the fixed and random effects. This formulation encompasses the modeling of data involving multiple sources of random error, such as repeated measures within subjects and clustered data collected from some experimental units (Breslow and Clayton, 1993).

We let the distribution for \mathbf{u} be $g(\mathbf{u}; \mathbf{D})$ that depends on parameters \mathbf{D} . The observed data \mathbf{y} are conditionally independent with density functions of the form

$$f(y_j|\mathbf{u}; \boldsymbol{\beta}, \kappa) = \exp [m_j \kappa^{-1} \{\theta_j y_j - b(\theta_j)\} + c(y_j; \kappa)] , \quad (5.23)$$

where θ_j is the canonical parameter, κ is the dispersion parameter, and m_j is the known prior weight. The conditional mean and canonical parameters are related through the equation $\mu_j = b'(\theta_j)$, where the prime denotes differentiation with respect to θ_j . Let $\boldsymbol{\Psi}$ denotes the vector of unknown parameters within $\boldsymbol{\beta}$, κ , and \mathbf{D} . The likelihood function for $\boldsymbol{\Psi}$ is given by

$$L(\boldsymbol{\Psi}) = \int \prod_{j=1}^n f(y_j|\mathbf{u}; \boldsymbol{\beta}, \kappa) g(\mathbf{u}; \mathbf{D}) d\mathbf{u} , \quad (5.24)$$

which cannot usually be evaluated in closed form and has an intractable integral whose dimension depends on the structure of the random effects.

Within the EM framework, the random effects are considered as missing data. The complete data is then $\mathbf{x} = (\mathbf{y}^\top, \mathbf{u}^\top)^\top$ and the complete-data log likelihood is given by

$$\log L_c(\boldsymbol{\Psi}) = \sum_{j=1}^n \log f(y_j|\mathbf{u}; \boldsymbol{\beta}, \kappa) + \log g(\mathbf{u}; \mathbf{D}) . \quad (5.25)$$

On the $(k+1)$ th iteration of the EM algorithm, the E-step involves the computation of the Q-function, $Q(\boldsymbol{\Psi}; \boldsymbol{\Psi}^{(k)}) = E_{\boldsymbol{\Psi}^{(k)}} \{\log L_c(\boldsymbol{\Psi})|\mathbf{y}\}$, where the expectation is with respect to the conditional distribution of $\mathbf{u}|\mathbf{y}$ with current parameter value $\boldsymbol{\Psi}^{(k)}$. As this conditional distribution involves the (marginal) likelihood function $L(\boldsymbol{\Psi})$ given in (5.24), an analytical evaluation of the Q-function for the model (5.23) will be impossible outside the normal theory mixed model (Booth and Hobert, 1999). The MCEM algorithm can be adopted to tackle this problem by replacing the expectation in the E-step with a MC approximation. Let $\mathbf{u}^{(1_k)}, \dots, \mathbf{u}^{(M_k)}$ denote a random sample from $k(\mathbf{u}|\mathbf{y}; \boldsymbol{\Psi}^{(k)})$ at the $(k+1)$ th iteration. A MC approximation of the Q-function is given by

$$Q_M(\boldsymbol{\Psi}; \boldsymbol{\Psi}^{(k)}) = \frac{1}{M} \sum_{m=1}^M \{\log f(\mathbf{y}|\mathbf{u}^{(m_k)}; \boldsymbol{\beta}, \kappa) + \log g(\mathbf{u}^{(m_k)}; \mathbf{D})\} . \quad (5.26)$$

From (5.26), it can be seen that the first term of the approximated Q -function involves only parameters β and κ , while the second term involves only D . Thus, the maximization in the MC M-step is usually relatively simple within the GLMM context (McCulloch, 1997).

Alternative simulation schemes for \mathbf{u} can be used for (5.26). For example, Booth and Hobert (1999) proposed the rejection sampling and a multivariate t importance sampling approximations. McCulloch (1997) considered dependent MC samples using MC Newton-Raphson (MCNR) algorithm.

5.4.2 Complicated M-Step

One of major reasons for the popularity of the EM algorithm is that the M-step involves only complete-data ML estimation, which is often computationally simple. But if the complete-data ML estimation is rather complicated, then the EM algorithm is less attractive. In many cases, however, complete-data ML estimation is relatively simple if maximization process on the M-step is undertaken conditional on some functions of the parameters under estimation. To this end, Meng and Rubin (1993) introduce a class of GEM algorithms, which they call the Expectation–Conditional Maximization (ECM) algorithm.

ECM and Multicycle ECM Algorithms

The ECM algorithm takes advantage of the simplicity of complete-data conditional maximization by replacing a complicated M-step of the EM algorithm with several computationally simpler conditional maximization (CM) steps. Each of these CM-steps maximizes the Q -function found in the preceding E-step subject to constraints on Ψ , where the collection of all constraints is such that the maximization is over the full parameter space of Ψ .

A CM-step might be in closed form or it might itself require iteration, but because the CM maximizations are over smaller dimensional spaces, often they are simpler, faster, and more stable than the corresponding full maximizations called for on the M-step of the EM algorithm, especially when iteration is required. The ECM algorithm typically converges more slowly than the EM in terms of number of iterations, but can be faster in total computer time. More importantly, the ECM algorithm preserves the appealing convergence properties of the EM algorithm, such as its monotone convergence.

We suppose that the M-step is replaced by $S > 1$ steps and let $\Psi^{(k+s/S)}$ denote the value of Ψ on the s th CM-step of the $(k+1)$ th iteration. In many applications of the ECM algorithm, the S CM-steps correspond to the situation where the parameter vector Ψ is partitioned into S subvectors,

$$\Psi = (\Psi_1^\top, \dots, \Psi_S^\top)^\top.$$

The s th CM-step then requires the maximization of the Q -function with respect to the s th subvector Ψ_s , with the other $(S-1)$ subvectors held fixed at their current values. The convergence properties and the rate of convergence of the ECM algo-

rithm have been discussed in Meng (1994) and Meng and Rubin (1993); see also the discussion in Sexton and Swensen (2000). In particular, it can be shown that

$$Q(\Psi^{(k+1)}; \Psi^{(k)}) \geq Q(\Psi^{(k+(S-1)/S)}; \Psi^{(k)}) \geq \dots \geq Q(\Psi^{(k)}; \Psi^{(k)}). \quad (5.27)$$

This shows that the ECM algorithm is a GEM algorithm and so possesses its desirable convergence properties. As noted in Sect. 5.2.3, the inequality (5.27) is a sufficient condition for

$$L(\Psi^{(k+1)}) \geq L(\Psi^{(k)}) \quad (5.28)$$

to hold.

In many cases, the computation of an E-step may be much cheaper than the computation of the CM-steps. Hence one might wish to perform one E-step before each CM-step. A cycle is defined to be one E-step followed by one CM-step. The corresponding algorithm is called the multicycle ECM (Meng and Rubin, 1993). A multicycle ECM may not necessarily be a GEM algorithm; that is, the inequality (5.27) may not hold. However, it is not difficult to show that the multicycle ECM algorithm monotonically increases the likelihood function $L(\Psi)$ after each cycle, and hence, after each iteration. The convergence results of the ECM algorithm apply to a multicycle version of it. An obvious disadvantage of using a multicycle ECM algorithm is the extra computation at each iteration. Intuitively, as a tradeoff, one might expect it to result in larger increases in the log likelihood function per iteration since the Q -function is being updated more often (Meng, 1994; Meng and Rubin, 1993).

Example 5: Single-Factor Analysis Model

Factor analysis is commonly used for explaining data, in particular, correlations between variables in multivariate observations and for dimensionality reduction. In a typical factor analysis model, each observation Y_j is modeled as

$$Y_j = \boldsymbol{\mu} + \mathbf{B}U_j + \mathbf{e}_j \quad (j = 1, \dots, n), \quad (5.29)$$

where U_j is a q -dimensional ($q < p$) vector of latent or unobservable variables called factors and \mathbf{B} is a $p \times q$ matrix of factor loadings (parameters). The U_j are assumed to be i.i.d. as $N(\mathbf{0}, \mathbf{I}_q)$, independently of the errors \mathbf{e}_j , which are assumed to be i.i.d. as $N(\mathbf{0}, \mathbf{D})$, where $\mathbf{D} = \text{diag}(\sigma_1^2, \dots, \sigma_p^2)$, and where \mathbf{I}_q denotes the $q \times q$ identity matrix. Thus, conditional on $U_j = \mathbf{u}_j$, the Y_j are independently distributed as $N(\boldsymbol{\mu} + \mathbf{B}\mathbf{u}_j, \mathbf{D})$. Unconditionally, the Y_j are i.i.d. according to a normal distribution with mean $\boldsymbol{\mu}$ and covariance matrix

$$\boldsymbol{\Sigma} = \mathbf{B}\mathbf{B}^\top + \mathbf{D}. \quad (5.30)$$

If q is chosen sufficiently smaller than p , representation (5.30) imposes some constraints on $\boldsymbol{\Sigma}$ and thus reduces the number of free parameters to be estimated. Note that in the case of $q > 1$, there is an infinity of choices for \mathbf{B} , since (5.30) is still satisfied if \mathbf{B} is replaced by $\mathbf{B}\mathbf{C}$, where \mathbf{C} is any orthogonal matrix of order q . As $\frac{1}{2}q(q-1)$ constraints are needed for \mathbf{B} to be uniquely defined, the number of

free parameters is $pq + p - \frac{1}{2}q(q-1)$; see Lawley and Maxwell (1971, Chap. 1) and McLachlan et al. (2003).

The factor analysis model (5.29) can be fitted by the EM algorithm and its variants. The MLE of the mean $\boldsymbol{\mu}$ is obviously the sample mean $\bar{\boldsymbol{y}}$ of the n observed values $\boldsymbol{y}_1, \dots, \boldsymbol{y}_n$ corresponding to the random sample $\boldsymbol{Y}_1, \dots, \boldsymbol{Y}_n$. Hence in the sequel, $\boldsymbol{\mu}$ can be set equal to $\bar{\boldsymbol{y}}$ without loss of generality. The log likelihood for $\boldsymbol{\Psi}$ that can be formed from the observed data $\boldsymbol{y} = (\boldsymbol{y}_1^\top, \dots, \boldsymbol{y}_n^\top)^\top$ is, apart from an additive constant,

$$\log L(\boldsymbol{\Psi}) = -\frac{1}{2}n \left\{ \log | \boldsymbol{B}\boldsymbol{B}^\top + \boldsymbol{D} | + \sum_{j=1}^n (\boldsymbol{y}_j - \boldsymbol{\mu})^\top (\boldsymbol{B}\boldsymbol{B}^\top + \boldsymbol{D})^{-1} (\boldsymbol{y}_j - \boldsymbol{\mu}) \right\}.$$

We follow Dempster et al. (1977) and formulate $\boldsymbol{x} = (\boldsymbol{y}^\top, \boldsymbol{u}_1^\top, \dots, \boldsymbol{u}_n^\top)^\top$ as the complete-data vector, where \boldsymbol{u}_j corresponds to \boldsymbol{U}_j . Thus, the complete-data log likelihood is, but for an additive constant,

$$\log L_c(\boldsymbol{\Psi}) = -\frac{1}{2}n \log | \boldsymbol{D} | - \frac{1}{2} \sum_{j=1}^n \left\{ (\boldsymbol{y}_j - \boldsymbol{\mu} - \boldsymbol{B}\boldsymbol{u}_j)^\top \boldsymbol{D}^{-1} (\boldsymbol{y}_j - \boldsymbol{\mu} - \boldsymbol{B}\boldsymbol{u}_j) + \boldsymbol{u}_j^\top \boldsymbol{u}_j \right\}.$$

The complete-data density belongs to the exponential family, and the complete-data sufficient statistics are C_{yy} , C_{yu} , and C_{uu} , where

$$C_{yy} = \sum_{j=1}^n (\boldsymbol{y}_j - \boldsymbol{\mu})(\boldsymbol{y}_j - \boldsymbol{\mu})^\top; \quad C_{yu} = \sum_{j=1}^n (\boldsymbol{y}_j - \boldsymbol{\mu})\boldsymbol{u}_j^\top; \quad C_{uu} = \sum_{j=1}^n \boldsymbol{u}_j\boldsymbol{u}_j^\top.$$

On the $(k+1)$ th iteration of the EM algorithm, we have

E-Step: Compute the conditional expectation of the sufficient statistics given \boldsymbol{y} and the current fit $\boldsymbol{\Psi}^{(k)}$ for $\boldsymbol{\Psi}$:

$$E_{\boldsymbol{\Psi}^{(k)}}(C_{yy} | \boldsymbol{y}) = C_{yy}, \quad E_{\boldsymbol{\Psi}^{(k)}}(C_{yu} | \boldsymbol{y}) = C_{yy}\boldsymbol{\gamma}^{(k)},$$

and

$$E_{\boldsymbol{\Psi}^{(k)}}(C_{uu} | \boldsymbol{y}) = \boldsymbol{\gamma}^{(k)\top} C_{yy}\boldsymbol{\gamma}^{(k)} + n\boldsymbol{\omega}^{(k)},$$

where

$$\boldsymbol{\gamma}^{(k)} = \left\{ \boldsymbol{B}^{(k)}\boldsymbol{B}^{(k)\top} + \boldsymbol{D}^{(k)} \right\}^{-1} \boldsymbol{B}^{(k)} \quad \text{and} \quad \boldsymbol{\omega}^{(k)} = \boldsymbol{I}_q - \boldsymbol{\gamma}^{(k)\top} \boldsymbol{B}^{(k)}.$$

M-Step: Calculate $\boldsymbol{B}^{(k+1)} = C_{yy}\boldsymbol{\gamma}^{(k)} \left(\boldsymbol{\gamma}^{(k)\top} C_{yy}\boldsymbol{\gamma}^{(k)} + n\boldsymbol{\omega}^{(k)} \right)^{-1}$ and

$$\begin{aligned} \boldsymbol{D}^{(k+1)} &= \text{diag} \left\{ C_{yy}/n - \boldsymbol{B}^{(k+1)}\boldsymbol{H}^{(k)}\boldsymbol{B}^{(k+1)\top} \right\} \\ &= n^{-1} \text{diag} \left\{ C_{yy} - C_{yy}\boldsymbol{\gamma}^{(k)}\boldsymbol{B}^{(k+1)\top} \right\}, \end{aligned} \quad (5.31)$$

where

$$\mathbf{H}^{(k)} = E_{\Psi^{(k)}}(\mathbf{C}_{uu} | \mathbf{y})/n = \boldsymbol{\gamma}^{(k)\top} \mathbf{C}_{yy} \boldsymbol{\gamma}^{(k)}/n + \boldsymbol{\omega}^{(k)}. \quad (5.32)$$

It is noted that direct differentiation of $\log L(\boldsymbol{\Psi})$ shows that the ML estimate of the diagonal matrix \mathbf{D} satisfies

$$\widehat{\mathbf{D}} = \text{diag} \left\{ \mathbf{C}_{yy}/n - \widehat{\mathbf{B}}\widehat{\mathbf{B}}^\top \right\}. \quad (5.33)$$

As remarked by Lawley and Maxwell (1971, pp. 30), (5.33) looks temptingly simple to use to solve for $\widehat{\mathbf{D}}$, but was not recommended due to convergence problems. On comparing (5.33) with (5.31), it can be seen that with the calculation of the ML estimate of \mathbf{D} directly from $\log L(\boldsymbol{\Psi})$, the unconditional expectation of $\mathbf{U}_j \mathbf{U}_j^\top$, which is the identity matrix, is used in place of the conditional expectation in (5.32) on the E-step. Although the EM algorithm is numerically stable, the rate of convergence is slow, which can be attributed to the typically large fraction of missing data. Liu and Rubin (1994, 1998) have considered the application of the ECME algorithm to this problem; see Sect. 5.4.3 for the description of the algorithm. The M-step is replaced by two CM-steps. On the first CM-step, $\mathbf{B}^{(k+1)}$ is calculated as on the M-step above, while on the second CM-step the diagonal matrix $\mathbf{D}^{(k+1)}$ is obtained by using an algorithm such as Newton–Raphson to maximize the actual log likelihood with \mathbf{B} fixed at $\mathbf{B}^{(k+1)}$.

The single-factor analysis model provides only a global linear model for the representation of the data in a lower-dimensional subspace, the scope of its application is limited. A global nonlinear approach can be obtained by postulating a mixture of linear submodels for the distribution of the full observation vector \mathbf{Y}_j given the (unobservable) factors \mathbf{u}_j (McLachlan et al., 2003). This mixture of factor analyzers has been adopted both (a) for model-based density estimation from high-dimensional data, and hence for the clustering of such data, and (b) for local dimensionality reduction; see for example McLachlan and Peel (2000, Chap. 8). Some more material on dimension reduction methods can be found in Chap. III.6 of this handbook.

Speeding up Convergence

5.4.3

Several suggestions are available in the literature for speeding up convergence, some of a general kind and some problem-specific; see for example McLachlan and Krishnan (1997, Chap. 4). Most of them are based on standard numerical analytic methods and suggest a hybrid of EM with methods based on Aitken acceleration, over-relaxation, line searches, Newton methods, conjugate gradients, etc. Unfortunately, the general behaviour of these hybrids is not always clear and they may not yield monotonic increases in the log likelihood over iterations. There are also methods that approach the problem of speeding up convergence in terms of “efficient” data augmentation scheme (Meng and van Dyk, 1997). Since the convergence rate of the EM algorithm increases with the proportion

of observed information in the prescribed EM framework (Sect. 5.2.4), the basic idea of the scheme is to search for an efficient way of augmenting the observed data. By efficient, they mean less augmentation of the observed data (greater speed of convergence) while maintaining the simplicity and stability of the EM algorithm. A common trade-off is that the resulting E- and/or M-steps may be made appreciably more difficult to implement. To this end, Meng and van Dyk (1997) introduce a working parameter in their specification of the complete data to index a class of possible schemes to facilitate the search.

ECME, AECM, and PX-EM Algorithms

Liu and Rubin (1994, 1998) present an extension of the ECM algorithm called the ECME (expectation–conditional maximization either) algorithm. Here the “either” refers to the fact that with this extension, each CM-step either maximizes the Q -function or the actual (incomplete-data) log likelihood function $\log L(\Psi)$, subject to the same constraints on Ψ . The latter choice should lead to faster convergence as no augmentation is involved. Typically, the ECME algorithm is more tedious to code than the ECM algorithm, but the reward of faster convergence is often worthwhile especially because it allows convergence to be more easily assessed.

A further extension of the EM algorithm, called the Space-Alternating Generalized EM (SAGE), has been proposed by Fessler and Hero (1994), where they update sequentially small subsets of parameters using appropriately smaller complete data spaces. This approach is eminently suitable for situations like image reconstruction where the parameters are large in number. Meng and van Dyk (1997) combined the ECME and SAGE algorithms. The so-called Alternating ECM (AECM) algorithm allows the data augmentation scheme to vary where necessary over the CM-steps, within and between iterations. With this flexible data augmentation and model reduction schemes, the amount of data augmentation decreases and hence efficient computations are achieved.

In contrast to the AECM algorithm where the optimal value of the working parameter is determined before EM iterations, a variant is considered by Liu et al. (1998) which maximizes the complete-data log likelihood as a function of the working parameter within each EM iteration. The so-called parameter-expanded EM (PX-EM) algorithm has been used for fast stable computation of MLE in a wide range of models. This variant has been further developed, known as the one-step-late PX-EM algorithm, to compute MAP or maximum penalized likelihood (MPL) estimates (van Dyk and Tang, 2003). Analogous convergence results hold for the ECME, AECM, and PX-EM algorithms as for the EM and ECM algorithms. More importantly, these algorithms preserve the monotone convergence of the EM algorithm as stated in (5.28).

Extensions to the EM for Data Mining Applications

With the computer revolution, massively huge data sets of millions of multidimensional observations are now commonplace. There is an ever increasing demand

on speeding up the convergence of the EM algorithm to large databases. But at the same time, it is highly desirable if its simplicity and stability can be preserved. In applications where the M-step is computationally simple, for example, in fitting multivariate normal mixtures, the rate of convergence of the EM algorithm depends mainly on the computation time of an E-step as each data point is visited at each E-step. There have been some promising developments on modifications to the EM algorithm for the ML fitting of mixture models to large databases that preserve the simplicity of implementation of the EM in its standard form.

Neal and Hinton (1998) proposed the incremental EM (IEM) algorithm to improve the convergence rate of the EM algorithm. With this algorithm, the available n observations are divided into B ($B \leq n$) blocks and the E-step is implemented for only a block of data at a time before performing a M-step. A “scan” of the IEM algorithm thus consists of B partial E-steps and B M-steps. The argument for improved rate of convergence is that the algorithm exploits new information more quickly rather than waiting for a complete scan of the data before parameters are updated by an M-step. Another method suggested by Neal and Hinton (1998) is the sparse EM (SPEM) algorithm. In fitting a mixture model to a data set by ML via the EM algorithm, the current estimates of some posterior probabilities $\tau_{ij}^{(k)}$ for a given data point y_j are often close to zero. For example, if $\tau_{ij}^{(k)} < 0.005$ for the first two components of a four-component mixture being fitted, then with the SPEM algorithm we would fix $\tau_{ij}^{(k)}$ ($i = 1, 2$) for membership of y_j with respect to the first two components at their current values and only update $\tau_{ij}^{(k)}$ ($i = 3, 4$) for the last two components. This sparse E-step will take time proportional to the number of components that needed to be updated. A sparse version of the IEM algorithm (SPIEM) can be formulated by combining the partial E-step and the sparse E-step. With these versions, the likelihood is still increased after each scan. Ng and McLachlan (2003a) study the relative performances of these algorithms with various number of blocks B for the fitting of normal mixtures. They propose to choose B to be that factor of n that is the closest to $B^* = \text{round}(n^{2/5})$ for unrestricted component-covariance matrices, where $\text{round}(r)$ rounds r to the nearest integer.

Other approaches for speeding up the EM algorithm for mixtures have been considered in Bradley et al. (1998) and Moore (1999). The former developed a scalable version of the EM algorithm to handle very large databases with a limited memory buffer. It is based on identifying regions of the data that are compressible and regions that must be maintained in memory. Moore (1999) has made use of multiresolution *kd*-trees (*mrkd*-trees) to speed up the fitting process of the EM algorithm on normal mixtures. Here *kd* stands for *k*-dimensional where, in our notation, $k = p$, the dimension of an observation y_j . His approach builds a multiresolution data structure to summarize the database at all resolutions of interest simultaneously. The *mrkd*-tree is a binary tree that recursively splits the whole set of data points into partitions. The contribution of all the data points in a tree node to the sufficient statistics is simplified by calculating at the mean of these data points to save time. Ng and McLachlan (2003b) combined the IEM algorithm with the *mrkd*-tree approach to further speed up the EM algorithm. They also studied

the convergence properties of this modified version and the relative performance with some other variants of the EM algorithm for speeding up the convergence for the fitting of normal mixtures.

Neither the scalable EM algorithm nor the *mrkd*-tree approach guarantee the desirable reliable convergence properties of the EM algorithm. Moreover, the scalable EM algorithm becomes less efficient when the number of components g is large, and the *mrkd*-trees-based algorithms slow down as the dimension p increases; see for example Ng and McLachlan (2003b) and the references therein. Further discussion on data mining applications can be found in Chap. III.13 of this handbook.

Miscellaneous Topics on the EM Algorithm

5.5

5.5.1 EM Algorithm for MAP Estimation

Although we have focussed on the application of the EM algorithm for computing MLEs in a frequentist framework, it can be equally applied to find the mode of the posterior distribution in a Bayesian framework. This problem is analogous to MLE and hence the EM algorithm and its variants can be adapted to compute MAP estimates. The computation of the MAP estimate in a Bayesian framework via the EM algorithm corresponds to the consideration of some prior density for Ψ . The E-step is effectively the same as for the computation of the MLE of Ψ in a frequentist framework, requiring the calculation of the Q-function. The M-step differs in that the objective function for the maximization process is equal to the Q-function, augmented by the log prior density. The combination of prior and sample information provides a posterior distribution of the parameter on which the estimation is based.

The advent of inexpensive high speed computers and the simultaneous rapid development in posterior simulation techniques such as Markov chain Monte Carlo (MCMC) methods (Gelfand and Smith, 1990) enable Bayesian estimation to be undertaken. In particular, posterior quantities of interest can be approximated through the use of MCMC methods such as the Gibbs sampler. Such methods allow the construction of an ergodic Markov chain with stationary distribution equal to the posterior distribution of the parameter of interest. A detailed description of the MCMC technology can be found in Chap. II.3.

Although the application of MCMC methods is now routine, there are some difficulties that have to be addressed with the Bayesian approach, particularly in the context of mixture models. One main hindrance is that improper priors yield improper posterior distributions. Another hindrance is that when the number of components g is unknown, the parameter space is simultaneously ill-defined and of infinite dimension. This prevents the use of classical testing procedures and priors (McLachlan and Peel, 2000, Chap. 4). A fully Bayesian approach with g

taken to be an unknown parameter has been considered by Richardson and Green (1997). Their MCMC methods allow *jumps* to be made for variable dimension parameters and thus can handle g being unspecified. A further hindrance is the effect of label switching, which arises when there is no real prior information that allows one to discriminate between the components of a mixture model belonging to the same parametric family. This effect is very important when the solution is being calculated iteratively and there is the possibility that the labels of the components may be switched on different iterations (McLachlan and Peel, 2000, Chap. 4).

Iterative Simulation Algorithms

5.5.2

In computing Bayesian solutions to incomplete-data problems, iterative simulation techniques have been adopted to find the MAP estimates or estimating the entire posterior density. These iterative simulation techniques are conceptually similar to the EM algorithm, simply replacing the E- and M-steps by draws from the current conditional distribution of the missing data and Ψ , respectively. However, in some methods such as the MCEM algorithm described in Sect. 5.4.1, only the E-step is so implemented. Many of these methods can be interpreted as iterative simulation analogs of the various versions of the EM and its extensions. Some examples are Stochastic EM, Data Augmentation algorithm, and MCMC methods such as the Gibbs sampler (McLachlan and Krishnan, 1997, Chap. 6). Here, we give a very brief outline of the Gibbs sampler; see also Chap. II.3 of this handbook and the references therein.

The Gibbs sampler is extensively used in many Bayesian problems where the joint distribution is too complicated to handle, but the conditional distributions are often easy enough to draw from; see Casella and George (1992). On the Gibbs sampler, an approximate sample from $p(\Psi | \mathbf{y})$ is obtained by simulating directly from the (full) conditional distribution of a subvector of Ψ given all the other parameters in Ψ and \mathbf{y} . We write $\Psi = (\Psi_1, \dots, \Psi_d)$ in component form, a d -dimensional Gibbs sampler makes a Markov transition from $\Psi^{(k)}$ to $\Psi^{(k+1)}$ via d successive simulations as follows:

- (1) Draw $\Psi_1^{(k+1)}$ from $p(\Psi_1 | \mathbf{y}; \Psi_2^{(k)}, \dots, \Psi_d^{(k)})$.
- (2) Draw $\Psi_2^{(k+1)}$ from $p(\Psi_2 | \mathbf{y}; \Psi_1^{(k+1)}, \Psi_3^{(k)}, \dots, \Psi_d^{(k)})$.
- \vdots
- (d) Draw $\Psi_d^{(k+1)}$ from $p(\Psi_d | \mathbf{y}; \Psi_1^{(k+1)}, \dots, \Psi_{d-1}^{(k+1)})$.

The vector sequence $\{\Psi^{(k)}\}$ thus generated is known to be a realization of a homogeneous Markov Chain. Many interesting properties of such a Markov sequence have been established, including geometric convergence, as $k \rightarrow \infty$; to a unique stationary distribution that is the posterior density $p(\Psi_1^{(k)}, \dots, \Psi_d^{(k)} | \mathbf{y})$ under certain conditions; see Roberts and Polson (1994). Among other sampling methods, there is the Metropolis-Hastings algorithm (Hastings, 1970), which, in contrast to

the Gibbs sampler, accepts the candidate simulated component in Ψ with some defined probability (McLachlan and Peel, 2000, Chap. 4).

The Gibbs sampler and other such iterative simulation techniques being Bayesian in their point of view consider both parameters and missing values as random variables and both are subjected to random draw operations. In the iterative algorithms under a frequentist framework, like the EM-type algorithms, parameters are subjected to a maximization operation and missing values are subjected to an averaging operation. Thus the various versions of the Gibbs sampler can be viewed as stochastic analogs of the EM, ECM, and ECME algorithms. Besides these connections, the EM-type algorithms also come in useful as starting points for iterative simulation algorithms where typically regions of high density are not known *a priori* (McLachlan and Krishnan, 1997, Sect. 6.7.3). The relationship between the EM algorithm and the Gibbs sampler and the connection between their convergence properties have been examined in Sahu and Roberts (1999).

5.5.3 Further Applications of the EM Algorithm

Since the publication of Dempster et al. (1977), the number, variety, and range of applications of the EM algorithm and its extensions have been tremendous. Applications in many different contexts can be found in monographs Little and Rubin (2002), McLachlan and Krishnan (1997), and McLachlan and Peel (2000). We conclude the chapter with a quick summary of some of the more interesting and topical applications of the EM algorithm.

Bioinformatics: Mixture of Factor Analyzers

The analysis of gene expression microarray data using clustering techniques has an important role to play in the discovery, validation, and understanding of various classes of cancer; see for example Alon et al. (1999) and van't Veer et al. (2002). Clustering algorithms can be applied to the problem of clustering genes and tumour tissues (McLachlan et al., 2002) and also in the discovery of motif patterns in DNA sequences (Bailey and Elkan, 1995); see also Chap. IV.3 for the description of biomolecular sequences and structures. The EM algorithm and its variants have been applied to tackle some of the problems arisen in such applications. For example, the clustering of tumour tissues on the basis of genes expression is a nonstandard cluster analysis problem since the dimension of each tissue sample is so much greater than the number of tissues. In McLachlan et al. (2002), mixture of factor analyzers is adopted to reduce effectively the dimension of the feature space of genes. The AECM algorithm (Meng and van Dyk, 1997) can be used to fit the mixture of factor analyzers by ML (McLachlan and Peel, 2000, Chap. 8).

Image Analysis: Hidden Markov Models

In image analysis, the observed data y_j refers to intensities measured on n pixels in a scene, the associated component indicator vectors z_j will not be independently distributed as the intensities between neighboring pixels are spatially correlated.

The set of hidden states z_j is viewed as missing data (McLachlan and Peel, 2000, Chap. 13; van Dyk and Meng, 2001) and a stationary Markovian model over a finite state space is generally formulated for the distribution of the hidden variable Z . In one dimension, this Markovian model is a Markov chain, and in two and higher dimensions a Markov random field (MRF) (Besag, 1986).

The use of the EM algorithm in a hidden Markov chain, known in the Hidden Markov model literature as the Baum-Welch algorithm (Baum et al., 1970), has been formulated long before Dempster et al. (1977). Also, Robert et al. (1993) consider a stochastic Bayesian approach to parameter estimation for a hidden Markov chain. Lystig and Hughes (2002) provide a means of implementing a NR approach to obtain parameter estimates and an exact computation of the observed information matrix for hidden Markov models.

The EM algorithm for the hidden MRF is considerably more difficult; see McLachlan (1992, Chap. 13) and the references therein. Even in the exponential family case (see Sect. 5.2.1) the E- and M-steps are difficult to perform even by numerical methods, except in some very simple cases like a one-parameter case; in some cases they may be implemented by suitable Gibbs sampler algorithms. A variety of practical procedures has been considered in the literature. They are reviewed by Qian and Titterton (1992), who also suggest a Monte Carlo restoration-estimation algorithm. An approximation to the E-step, based on a fractional weight version of Besag's iterated conditional modes (ICM) algorithm (Besag, 1986), has been adopted for the segmentation of magnetic resonance images (McLachlan and Peel, 2000, Sect. 13.4). An alternative approach is a Bayesian one, where the likelihood can be regularized using a prior, resulting in a better-conditioned log likelihood. This can also be interpreted as a penalized likelihood approach. Random field models such as Gibbs priors are often used in this context to capture the local smooth structures of the images (Geman and Geman, 1984).

References

- Alon, U., Barkai, N., Notterman, D.A. et al. (1999). Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays. *Proceedings of the National Academy of Sciences USA*, 96:6745–6750.
- Bailey, T.L. and Elkan, C. (1995). Unsupervised learning of multiple motifs in biopolymers using expectation maximization. *Machine Learning*, 21:51–80.
- Baker, S.G. (1992). A simple method for computing the observed information matrix when using the EM algorithm with categorical data. *Journal of Computational and Graphical Statistics*, 1:63–76.
- Basford, K.E., Greenway, D.R., McLachlan, G.J., and Peel, D. (1997). Standard errors of fitted means under normal mixture models. *Computational Statistics*, 12:1–17.
- Baum, L.E., Petrie, T., Soules, G., and Weiss, N. (1970). A maximisation technique occurring in the statistical analysis of probabilistic functions of Markov processes. *Annals of Mathematical Statistics*, 41:164–171.

- Besag, J. (1986). On the statistical analysis of dirty pictures. *Journal of the Royal Statistical Society, Series B*, 48:259–302.
- Booth, J.G. and Hobert, J.P. (1999). Maximizing generalized linear mixed model likelihoods with an automated Monte Carlo EM algorithm. *Journal of the Royal Statistical Society, Series B*, 61:265–285.
- Bradley, P.S., Fayyad, U.M., and Reina, C.A. (1998). Scaling EM (expectation-maximization) clustering to large databases. Technical Report No. MSR-TR-98-35 (revised February, 1999), Microsoft Research, Seattle.
- Breslow, N.E. and Clayton, D.G. (1993). Approximate inference in generalized linear mixed models. *Journal of the American Statistical Association*, 88:9–25.
- Casella, G. and George, E.I. (1992). Explaining the Gibbs sampler. *American Statistician*, 46:167–174.
- Cramér, H. (1946). *Mathematical Methods of Statistics*. Princeton University Press, Princeton, New Jersey.
- Dempster, A.P., Laird, N.M., and Rubin, D.B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B*, 39:1–38.
- Efron, B. (1979). Bootstrap methods: another look at the jackknife. *Annals of Statistics*, 7:1–26.
- Efron, B. and Tibshirani, R. (1993). *An Introduction to the Bootstrap*. Chapman & Hall, London.
- Fessler, J.A. and Hero, A.O. (1994). Space-alternating generalized expectation-maximization algorithm. *IEEE Transactions on Signal Processing*, 42:2664–2677.
- Flury, B. and Zoppé, A. (2000). Exercises in EM. *American Statistician*, 54:207–209.
- Gelfand, A.E. and Smith, A.F.M. (1990). Sampling-based approaches to calculating marginal densities. *Journal of the American Statistical Association*, 85:398–409.
- Geman, S. and Geman, D. (1984). Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6:721–741.
- Hastings, W.K. (1970). Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, 57:97–109.
- Jamshidian, M. and Jennrich, R.I. (2000). Standard errors for EM estimation. *Journal of the Royal Statistical Society, Series B*, 62:257–270.
- Lawley, D.N. and Maxwell, A.E. (1971). *Factor Analysis as a Statistical Method*. Butterworths, London, second edition.
- Little, R.J.A. and Rubin, D.B. (2002). *Statistical Analysis with Missing Data*. Wiley, New York, second edition.
- Liu, C. and Rubin, D.B. (1994). The ECME algorithm: a simple extension of EM and ECM with faster monotone convergence. *Biometrika*, 81:633–648.
- Liu, C. and Rubin, D.B. (1998). Maximum likelihood estimation of factor analysis using the ECME algorithm with complete and incomplete data. *Statistica Sinica*, 8:729–747.

- Liu, C., Rubin, D.B., and Wu, Y.N. (1998). Parameter expansion to accelerate EM: the PX-EM algorithm. *Biometrika*, 85:755–770.
- Louis, T.A. (1982). Finding the observed information matrix when using the EM algorithm. *Journal of the Royal Statistical Society, Series B*, 44:226–233.
- Lystig, T.C. and Hughes, J.P. (2002). Exact computation of the observed information matrix for hidden Markov models. *Journal of Computational and Graphical Statistics*, 11:678–689.
- McCullagh, P.A. and Nelder, J. (1989). *Generalized Linear Models*. Chapman & Hall, London, second edition.
- McCulloch, C.E. (1997). Maximum likelihood algorithms for generalized linear mixed models. *Journal of the American Statistical Association*, 92:162–170.
- McLachlan, G.J. (1992). *Discriminant Analysis and Statistical Pattern Recognition*. Wiley, New York.
- McLachlan, G.J. and Basford, K.E. (1988). *Mixture Models: Inference and Applications to Clustering*. Marcel Dekker, New York.
- McLachlan, G.J., Bean, R.W., and Peel, D. (2002). A mixture model-based approach to the clustering of microarray expression data. *Bioinformatics*, 18:413–422.
- McLachlan, G.J. and Krishnan, T. (1997). *The EM Algorithm and Extensions*. Wiley, New York.
- McLachlan, G.J. and Peel, D. (2000). *Finite Mixture Models*. Wiley, New York.
- McLachlan, G.J., Peel, D., and Bean, R.W. (2003). Modelling high-dimensional data by mixtures of factor analyzers. *Computational Statistics & Data Analysis*, 41:379–388.
- Meilijson, I. (1989). A fast improvement of the EM algorithm in its own terms. *Journal of the Royal Statistical Society, Series B*, 51:127–138.
- Meng, X.L. (1994). On the rate of convergence of the ECM algorithm. *Annals of Statistics*, 22:326–339.
- Meng, X.L. and Rubin, D.B. (1991). Using EM to obtain asymptotic variance-covariance matrices: the SEM algorithm. *Journal of the American Statistical Association*, 86:899–909.
- Meng, X.L. and Rubin, D.B. (1993). Maximum likelihood estimation via the ECM algorithm: a general framework. *Biometrika*, 80:267–278.
- Meng, X.L. and van Dyk, D. (1997). The EM algorithm – an old folk song sung to a fast new tune. *Journal of the Royal Statistical Society, Series B*, 59:511–567.
- Moore, A.W. (1999). Very fast EM-based mixture model clustering using multiresolution kd-trees. In Kearns, M.S., Solla, S.A., and Cohn, D.A., editors, *Advances in Neural Information Processing Systems 11*, pages 543–549. MIT Press, MA.
- Neal, R.M. and Hinton, G.E. (1998). A view of the EM algorithm that justifies incremental, sparse, and other variants. In Jordan, M.I., editor, *Learning in Graphical Models*, pages 355–368. Kluwer, Dordrecht.
- Nettleton, D. (1999). Convergence properties of the EM algorithm in constrained parameter spaces. *Canadian Journal of Statistics*, 27:639–648.
- Newton, M.A. and Raftery, A.E. (1994). Approximate Bayesian inference with the weighted likelihood bootstrap. *Journal of the Royal Statistical Society, Series B*, 56:3–48.

- Ng, S.K. and McLachlan, G.J. (2003a). On the choice of the number of blocks with the incremental EM algorithm for the fitting of normal mixtures. *Statistics and Computing*, 13:45–55.
- Ng, S.K. and McLachlan, G.J. (2003b). On some variants of the EM algorithm for the fitting of finite mixture models. *Austrian Journal of Statistics*, 32:143–161.
- Qian, W. and Titterton, D.M. (1992). Stochastic relaxations and EM algorithms for Markov random fields. *Journal of Statistical Computation and Simulation*, 40:55–69.
- Richardson, S. and Green, P.J. (1997). On Bayesian analysis of mixtures with an unknown number of components. *Journal of the Royal Statistical Society, Series B*, 59:731–792 (correction (1998), pp. 661).
- Robert, C.P., Celeux, G., and Diebolt, J. (1993). Bayesian estimation of hidden Markov chains: A stochastic implementation. *Statistics & Probability Letters*, 16:77–83.
- Roberts, G.O. and Polson, N.G. (1994). On the geometric convergence of the Gibbs sampler. *Journal of the Royal Statistical Society, Series B*, 56:377–384.
- Sahu, S.K. and Roberts, G.O. (1999). On convergence of the EM algorithm and the Gibbs sampler. *Statistics and Computing*, 9:55–64.
- Sexton, J. and Swensen, A.R. (2000). ECM algorithms that converge at the rate of EM. *Biometrika*, 87:651–662.
- Ueda, N. and Nakano, R. (1998). Deterministic annealing EM algorithm. *Neural Networks*, 11:271–282.
- van Dyk, D.A. and Meng, X.L. (2001). The art of data augmentation. *Journal of Computational and Graphical Statistics*, 10:1–111.
- van Dyk, D.A. and Tang, R. (2003). The one-step-late PXEM algorithm. *Statistics and Computing*, 13:137–152.
- van't Veer, L.J., Dai, H., van de Vijver, M.J. et al. (2002). Gene expression profiling predicts clinical outcome of breast cancer. *Nature*, 415:530–536.
- Wei, G.C.G. and Tanner, M.A. (1990). A Monte Carlo implementation of the EM algorithm and the poor man's data augmentation algorithms. *Journal of the American Statistical Association*, 85:699–704.
- Wright, K. and Kennedy, W.J. (2000). An interval analysis approach to the EM algorithm. *Journal of Computational and Graphical Statistics*, 9:303–318.
- Wu, C.F.J. (1983). On the convergence properties of the EM algorithm. *Annals of Statistics*, 11:95–103.

Stochastic Optimization

II.6

James C. Spall

6.1	<i>Introduction</i>	170
	General Background	170
	Formal Problem Statement	171
	Contrast of Stochastic and Deterministic Optimization	172
	Some Principles of Stochastic Optimization	174
6.2	<i>Random Search</i>	176
	Some General Properties of Direct Random Search	176
	Two Algorithms for Random Search	177
6.3	<i>Stochastic Approximation</i>	180
	Introduction	180
	Finite-Difference SA	181
	Simultaneous Perturbation SA	183
6.4	<i>Genetic Algorithms</i>	186
	Introduction	186
	Chromosome Coding and the Basic GA Operations	188
	The Core Genetic Algorithm	191
	Some Implementation Aspects	191
	Some Comments on the Theory for GAs	192
6.5	<i>Concluding Remarks</i>	194

Stochastic optimization algorithms have been growing rapidly in popularity over the last decade or two, with a number of methods now becoming “industry standard” approaches for solving challenging optimization problems. This paper provides a synopsis of some of the critical issues associated with stochastic optimization and a gives a summary of several popular algorithms. Much more complete discussions are available in the indicated references.

To help constrain the scope of this article, we restrict our attention to methods using only measurements of the criterion (loss function). Hence, we do not cover the many stochastic methods using information such as gradients of the loss function. Section 6.1 discusses some general issues in stochastic optimization. Section 6.2 discusses random search methods, which are simple and surprisingly powerful in many applications. Section 6.3 discusses stochastic approximation, which is a foundational approach in stochastic optimization. Section 6.4 discusses a popular method that is based on connections to natural evolution – genetic algorithms. Finally, Sect. 6.5 offers some concluding remarks.

6.1 Introduction

6.1.1 General Background

Stochastic optimization plays a significant role in the analysis, design, and operation of modern systems. Methods for stochastic optimization provide a means of coping with inherent system noise and coping with models or systems that are highly nonlinear, high dimensional, or otherwise inappropriate for classical deterministic methods of optimization. Stochastic optimization algorithms have broad application to problems in statistics (e.g., design of experiments and response surface modeling), science, engineering, and business. Algorithms that employ some form of stochastic optimization have become widely available. For example, many modern data mining packages include methods such as simulated annealing and genetic algorithms as tools for extracting patterns in data.

Specific applications include business (making short- and long-term investment decisions in order to increase profit), aerospace engineering (running computer simulations to refine the design of a missile or aircraft), medicine (designing laboratory experiments to extract the maximum information about the efficacy of a new drug), and traffic engineering (setting the timing for the signals in a traffic network). There are, of course, *many* other applications.

Let us introduce some concepts and notation. Suppose Θ is the domain of allowable values for a vector θ . The fundamental problem of interest is to find the value(s) of a vector $\theta \in \Theta$ that minimize a scalar-valued *loss function* $L(\theta)$. Other common names for L are *performance measure*, *objective function*, *measure-of-effectiveness (MOE)*, *fitness function* (or *negative fitness function*), or *criterion*. While this problem refers to *minimizing* a loss function, a maximization problem (e.g., maximizing profit) can be trivially converted to a minimization problem by

changing the sign of the criterion. This paper focuses on the problem of minimization. In some cases (i.e., differentiable L), the minimization problem can be converted to a root-finding problem of finding θ such that $\mathbf{g}(\theta) = \partial L(\theta)/\partial \theta = \mathbf{0}$. Of course, this conversion must be done with care because such a root may not correspond to a global minimum of L .

The three remaining subsections in this section define some basic quantities, discuss some contrasts between (classical) deterministic optimization and stochastic optimization, and discuss some basic properties and fundamental limits. This section provides the foundation for interpreting the algorithm presentations in Sects. 6.2 to 6.4. There are many other references that give general reviews of various aspects of stochastic optimization. Among these are Arsham (1998), Fouskakis and Draper (2002), Fu (2002), Gosavi (2003), Michalewicz and Fogel (2000), and Spall (2003).

Formal Problem Statement

6.1.2

The problem of minimizing a loss function $L = L(\theta)$ can be formally represented as finding the set:

$$\Theta^* \equiv \arg \min_{\theta \in \Theta} L(\theta) = \{\theta^* \in \Theta : L(\theta^*) \leq L(\theta) \text{ for all } \theta \in \Theta\}, \quad (6.1)$$

where θ is the p -dimensional vector of parameters that are being adjusted and $\Theta \subseteq \mathbb{R}^p$. The “ $\arg \min_{\theta \in \Theta}$ ” statement in (6.1) should be read as: Θ^* is the set of values $\theta = \theta^*$ (θ the “argument” in “arg min”) that minimize $L(\theta)$ subject to θ^* satisfying the constraints represented in the set Θ . The elements $\theta^* \in \Theta^* \subseteq \Theta$ are equivalent solutions in the sense that they yield identical values of the loss function. The solution set Θ^* in (6.1) may be a unique point, a countable (finite or infinite) collection of points, or a set containing an uncountable number of points.

For ease of exposition, this paper generally focuses on continuous optimization problems, although some of the methods may also be used in discrete problems. In the continuous case, it is often assumed that L is a “smooth” (perhaps several times differentiable) function of θ . Continuous problems arise frequently in applications such as model fitting (parameter estimation), adaptive control, neural network training, signal processing, and experimental design. Discrete optimization (or *combinatorial optimization*) is a large subject unto itself (resource allocation, network routing, policy planning, etc.).

A major issue in optimization is distinguishing between global and local optima. All other factors being equal, one would always want a globally optimal solution to the optimization problem (i.e., at least one θ^* in the set of values Θ^*). In practice, however, it may not be feasible to find a global solution and one must be satisfied with obtaining a *local* solution. For example, L may be shaped such that there is a clearly defined minimum point over a broad region of the domain Θ , while there is a very narrow spike at a distant point. If the trough of this spike is lower than any point in the broad region, the local optimal solution is better than any nearby θ , but it is not the best possible θ .

It is usually only possible to ensure that an algorithm will approach a local minimum with a finite amount of resources being put into the optimization process. That is, it is easy to construct functions that will “fool” any known algorithm, unless the algorithm is given explicit prior information about the location of the global solution – certainly not a case of practical interest! However, since the local minimum may still yield a significantly improved solution (relative to no formal optimization process at all), the local minimum may be a fully acceptable solution for the resources available (human time, money, computer time, etc.) to be spent on the optimization. However, we discuss several algorithms (random search, stochastic approximation, and genetic algorithms) that are *sometimes* able to find global solutions from among multiple local solutions.

6.1.3 Contrast of Stochastic and Deterministic Optimization

As a paper on *stochastic* optimization, the algorithms considered here apply where:

- I. There is random noise in the measurements of $L(\theta)$
 - and/or –
- II. There is a random (Monte Carlo) choice made in the search direction as the algorithm iterates toward a solution.

In contrast, classical deterministic optimization assumes that perfect information is available about the loss function (and derivatives, if relevant) and that this information is used to determine the search direction in a deterministic manner at every step of the algorithm. In many practical problems, such information is not available. We discuss properties I and II below.

Let $\hat{\theta}_k$ be the generic notation for the estimate for θ at the k th iteration of whatever algorithm is being considered, $k = 0, 1, 2, \dots$. Throughout this paper, the *specific* mathematical form of $\hat{\theta}_k$ will change as the algorithm being considered changes. The following notation will be used to represent noisy measurements of L at a specific θ :

$$y(\theta) \equiv L(\theta) + \varepsilon(\theta), \quad (6.2)$$

where ε represents the noise term. Note that the noise terms show dependence on θ . This dependence is relevant for many applications. It indicates that the common statistical assumption of independent, identically distributed (i.i.d.) noise does not necessarily apply since θ will be changing as the search process proceeds.

Relative to property I, noise fundamentally alters the search and optimization process because the algorithm is getting potentially misleading information throughout the search process. For example, as described in Example 1.4 of Spall (2003), consider the following loss function with a scalar θ : $L(\theta) = e^{-0.1\theta} \sin(2\theta)$. If the domain for optimization is $\Theta = [0, 7]$, the (unique) minimum occurs at $\theta^* = 3\pi/4 \approx 2.36$, as shown in Fig. 6.1. Suppose that the analyst carrying out the optimization is not able to calculate $L(\theta)$, obtaining instead only *noisy* measurements $y(\theta) = L(\theta) + \varepsilon$, where the noises ε are i.i.d. with distribution $N(0, 0.5^2)$

(a normal distribution with mean zero and variance 0.5^2). The analyst uses the $y(\theta)$ measurements in conjunction with an algorithm to attempt to find θ^* .

Consider the experiment depicted in Fig. 6.1 (with data generated via MATLAB). Based on the simple method of collecting one measurement at each increment of 0.1 over the interval defined by Θ (including the endpoints 0 and 7), the analyst would falsely conclude that the minimum is at $\theta = 5.9$. As shown, this false minimum is far from the actual θ^* .

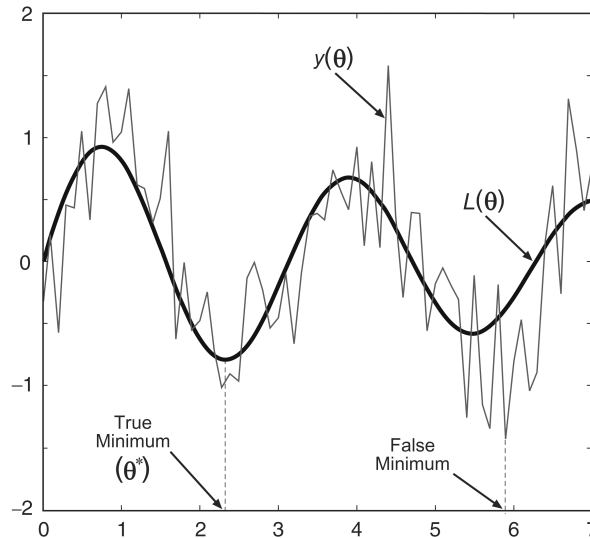


Figure 6.1. Simple loss function $L(\theta)$ with indicated minimum θ^* . Note how noise causes the algorithm to be deceived into sensing that the minimum is at the indicated false minimum (Reprinted from *Introduction to Stochastic Search and Optimization* with permission of John Wiley & Sons, Inc.)

Noise in the loss function measurements arises in almost any case where physical system measurements or computer simulations are used to approximate a steady-state criterion. Some specific areas of relevance include real-time estimation and control problems where data are collected “on the fly” as a system is operating and problems where large-scale simulations are run as estimates of actual system behavior.

Let us summarize two distinct problems involving noise in the loss function measurements: target tracking and simulation-based optimization. In the tracking problem there is a mean-squared error (MSE) criterion of the form

$$L(\theta) = E \left(\|\text{actual output} - \text{desired output}\|^2 \right) .$$

The stochastic optimization algorithm uses the actual (observed) squared error $y(\theta) = \|\cdot\|^2$, which is equivalent to an observation of L embedded in noise. In the simulation problem, let $L(\theta)$ be the loss function representing some type of

“average” performance for the system. A single run of a Monte Carlo simulation at a specific value of θ provides a noisy measurement: $y(\theta) = L(\theta) + \text{noise at } \theta$. (Note that it is rarely desirable to spend computational resources in averaging many simulation runs at a given value of θ ; in optimization, it is typically necessary to consider many values of θ .) The above problems are described in more detail in Examples 1.5 and 1.6 in Spall (2003).

Relative to the other defining property of stochastic optimization, property II (i.e., randomness in the search direction), it is sometimes beneficial to deliberately introduce randomness into the search process as a means of speeding convergence and making the algorithm less sensitive to modeling errors. This injected (Monte Carlo) randomness is usually created via computer-based pseudorandom number generators. One of the roles of injected randomness in stochastic optimization is to allow for “surprise” movements to unexplored areas of the search space that may contain an unexpectedly good θ value. This is especially relevant in seeking out a global optimum among multiple local solutions. Some algorithms that use injected randomness are random search (Sect. 6.2), simultaneous perturbation stochastic approximation (Sect. 6.3), and genetic algorithms (Sect. 6.4).

6.1.4 Some Principles of Stochastic Optimization

The discussion above is intended to motivate some of the issues and challenges in stochastic optimization. Let us now summarize some important issues for the implementation and interpretation of results in stochastic optimization.

The first issue we mention is the fundamental limits in optimization with only noisy information about the L function. Foremost, perhaps, is that the statistical error of the information fed into the algorithm – and the resulting error of the output of the algorithm – can only be reduced by incurring a significant cost in number of function evaluations. For the simple case of independent noise, the error decreases at the rate $1/\sqrt{N}$, where N represents the number of L measurements fed into the algorithm. This is a classical result in statistics, indicating that a 25-fold increase in function evaluations reduces the error by a factor of five.

A further limit for multivariate ($p > 1$) optimization is that the volume of the search region generally grows rapidly with dimension. This implies that one must usually exploit problem structure to have a hope of getting a reasonable solution in a high-dimensional problem.

All practical problems involve at least some restrictions on θ , although in some applications it may be possible to effectively ignore the constraints. Constraints can be encountered in many different ways, as motivated by the specific application. Note that the constraint set Θ does not necessarily correspond to the set of allowable values for θ in the search since some problems allow for the “trial” values of the search to be outside the set of allowable final estimates. Constraints are usually handled in practice on an ad hoc basis, especially tuned to the problem at hand. There are few general, practical methods that apply broadly in stochastic optimization. Michalewicz and Fogel (2000, Chap. 9), for

example, discuss some of the practical methods by which constraints are handled in evolutionary computation. Similar methods apply in other stochastic algorithms.

In general search and optimization, it is very difficult (perhaps impossible) to develop automated methods for indicating when the algorithm is close enough to the solution that it can be stopped. Without prior knowledge, there is always the possibility that θ^* could lie in some unexplored region of the search space. This applies even when the functions involved are relatively benign; see Solis and Wets (1981) for mention of this in the context of twice-differentiable convex L . Difficulties are compounded when the function measurements include noise.

It is quite normal for the environment to change over time. Hence, the solution to a problem now may not be the best (or even a good) solution to the corresponding problem in the future. In some search and optimization problems, the algorithm will be explicitly designed to adapt to a changing environment and automatically provide a new estimate at the optimal value (e.g., a control system). In other cases, one needs to restart the process and find a new solution. In either sense, the problem solving may never stop!

In reading or contributing to the literature on stochastic optimization, it is important to recognize the limits of numerical comparisons by Monte Carlo. Monte Carlo studies can be a sound scientific method of gaining insight and can be a useful supplement to theory, much of which is based on asymptotic (infinite sample) analysis. In fact, it is especially popular in certain branches of optimization to create “test suites” of problems, where various algorithms compete against each other. A danger arises, however, in making *broad* claims about the performance of individual algorithms based on the results of numerical studies. Performance can vary tremendously under even small changes in the form of the functions involved or the coefficient settings within the algorithms themselves. One must be careful about drawing conclusions beyond those directly supported by the specific numerical studies performed. For purposes of drawing objective conclusions about the relative performance of algorithms, it is preferable to use *both* theory and numerical studies.

Some real systems have one (unique) globally “best” operating point (θ^*) in the domain Θ while others have multiple global solutions (in either case, of course, there could be many *locally* optimal solutions). To avoid excessively cumbersome discussion of algorithms and supporting implementation issues and theory, we will often refer to “the” solution θ^* (versus “a” solution θ^*). In practice, an analyst may be quite satisfied to reach a solution at or close to *any* one $\theta^* \in \Theta^*$.

The so-called *no free lunch* (NFL) theorems provide a formal basis for the intuitively appealing idea that there is a fundamental tradeoff between algorithm efficiency and algorithm robustness (reliability and stability in a broad range of problems). In essence, algorithms that are very efficient on one type of problem are not automatically efficient on problems of a different type. Hence, there can never be a universally best search algorithm just as there is rarely (never?) a universally best solution to any general problem of society. Wolpert and Macready (1997)

provided a general formal structure for the NFL theorems, although the general ideas had been around for a long time prior to their paper (Wolpert and Macready were the ones to coin the expression “no free lunch” in this search and optimization context). The NFL theorems are established for discrete optimization with a finite (but arbitrarily large) number of options. However, their applicability includes most practical continuous problems because virtually all optimization is carried out on 32- or 64-bit digital computers. The theorems apply to the cases of both noise-free and noisy loss measurements. NFL states, in essence, that an algorithm that is effective on one class of problems is *guaranteed* to be ineffective on another class. Spall (2003, Sects. 1.2.2 and 10.6) provides more-detailed discussion on the basis and implications of NFL.

We are now in a position to discuss several popular stochastic optimization methods. The summaries here are just that – summaries. Much more complete discussions are available in the indicated references or in Spall (2003). We let $\hat{\theta}_k$ represent the estimate for θ at the k th iteration of an algorithm under consideration. Section 6.2 discusses random search methods, which are simple and surprisingly powerful in many applications. Section 6.3 discusses stochastic approximation and Sect. 6.4 discusses the popular genetic algorithms. Because of the relative brevity of this review, there are many methods of stochastic optimization not covered here, including simulated annealing, stochastic programming, evolutionary computation other than genetic algorithms, temporal difference methods, and so on. Readers with an interest in one of those may consult the references listed at the end of Sect. 6.1.1.

6.2 Random Search

This section describes some simple methods based on the notion of randomly searching over the domain of interest. Section 6.2.1 gives a short discussion of general issues in direct random search methods. The algorithms discussed in Sect. 6.2.2 represent two versions of random search.

6.2.1 Some General Properties of Direct Random Search

Consider the problem of trying to find the optimal $\theta \in \Theta$ based on noise-free measurements of $L = L(\theta)$. Random search methods are perhaps the simplest methods of stochastic optimization in such a setting and can be quite effective in many problems. Their relative simplicity is an appealing feature to both practitioners and theoreticians. These direct random search methods have a number of advantages relative to most other search methods. The advantages include relative ease of coding in software, the need to only obtain L measurements (versus gradients or other ancillary information), reasonable computational efficiency (especially for those direct search algorithms that make use of some local information in their search), broad applicability to non-trivial loss functions and/or to θ that may be

continuous, discrete, or some hybrid form, and a strong theoretical foundation. Some of these attributes were mentioned in the forward-looking paper of Karnopp (1963). A good recent survey of random search and related methods is Kolda et al. (2003).

Two Algorithms for Random Search

6.2.2

This section describes two direct random search techniques. These two algorithms represent only a tiny fraction of available methods. Solis and Wets (1981) and Zhigljavsky (1991) are among many references discussing these and other random search methods. The two algorithms here are intended to convey the essential flavor of most available direct random search algorithms. With the exception of some discussion at the end of the subsection, the methods here assume perfect (noise-free) values of L .

The first method we discuss is “blind random search.” This is the simplest random search method, where the current sampling for θ does not take into account the previous samples. That is, this blind search approach does not adapt the current sampling strategy to information that has been garnered in the search process. The approach can be implemented in batch (non-recursive) form simply by laying down a number of points in Θ and taking the value of θ yielding the lowest L value as our estimate of the optimum. The approach can be conveniently implemented in recursive form as we illustrate below.

The simplest setting for conducting the random sampling of new (candidate) values of θ is when Θ is a hypercube and we are using uniformly generated values of θ . The uniform distribution is continuous or discrete for the elements of θ depending on the definitions for these elements. *In fact, the blind search form of the algorithm is unique among all general stochastic optimization algorithms in that it is the only one without any adjustable algorithm coefficients that need to be “tuned” to the problem at hand.* (Of course, a de facto tuning decision has been made by choosing the uniform distribution for sampling.)

For a domain Θ that is not a hypercube or for other sampling distributions, one may use transformations, rejection methods, or Markov chain Monte Carlo to generate the sample θ values (see, e.g., Gentle, 2003). For example, if Θ is an irregular shape, one can generate a sample on a hypercube superset containing Θ and then reject the sample point if it lies outside of Θ .

The steps for a recursive implementation of blind random search are given below. This method applies when θ has continuous, discrete, or hybrid elements.

Blind Random Search

- Step 0 (Initialization)* Choose an initial value of θ , say $\hat{\theta}_0 \in \Theta$, either randomly or deterministically. (If random, usually a uniform distribution on Θ is used.) Calculate $L(\hat{\theta}_0)$. Set $k = 0$.
- Step 1* Generate a new independent value $\theta_{\text{new}}(k+1) \in \Theta$, according to the chosen probability distribution. If $L(\theta_{\text{new}}(k+1)) < L(\hat{\theta}_k)$, set $\hat{\theta}_{k+1} = \theta_{\text{new}}(k+1)$. Else, take $\hat{\theta}_{k+1} = \hat{\theta}_k$.

Step 2 Stop if the maximum number of L evaluations has been reached or the user is otherwise satisfied with the current estimate for θ via appropriate stopping criteria; else, return to Step 1 with the new k set to the former $k + 1$.

The above algorithm converges almost surely (a.s.) to θ^* under very general conditions (see, e.g., Spall, 2003, pp. 40–41). Of course, convergence alone is an incomplete indication of the performance of the algorithm. It is also of interest to examine the *rate* of convergence. The rate is intended to tell the analyst how close $\hat{\theta}_k$ is likely to be to θ^* for a given cost of search. While blind random search is a reasonable algorithm when θ is low dimensional, it can be shown that the method is generally a very slow algorithm for even moderately dimensioned θ (see, e.g., Spall, 2003, 42–43). This is a direct consequence of the exponential increase in the size of the search space as p increases. As an illustration, Spall (2003, Example 2.2) considers a case where $\Theta = [0, 1]^p$ (the p -dimensional hypercube with minimum and maximum values of 0 and 1 for each component of θ) and where one wishes to guarantee with probability 0.90 that each element of θ is within 0.04 units of the optimal value. As p increases from one to ten, there is an approximate 10^{10} -fold increase in the number of loss function evaluations required.

Blind search is the simplest random search in that the sampling generating the new θ value does not take account of where the previous estimates of θ have been. The random search algorithm below is slightly more sophisticated in that the random sampling is a function of the position of the current best estimate for θ . In this way, the search is more localized in the neighborhood of that estimate, allowing for a better exploitation of information that has previously been obtained about the shape of the loss function.

The localized algorithm is presented below. This algorithm was described in Matyas (1965). Note that the use of the term “localized” here pertains to the sampling strategy and does not imply that the algorithm is only useful for local (versus global) optimization in the sense described in Sect. 6.1. In fact, the algorithm has global convergence properties as described below. As with blind search, the algorithm may be used for continuous or discrete problems.

Localized Random Search

Step 0 (Initialization) Pick an initial guess $\hat{\theta}_0 \in \Theta$, either randomly or with prior information. Set $k = 0$.

Step 1 Generate an independent random vector $d_k \in \mathbb{R}^p$ and add it to the current θ value, $\hat{\theta}_k$. Check if $\hat{\theta}_k + d_k \in \Theta$. If $\hat{\theta}_k + d_k \notin \Theta$, generate a new d_k and repeat or, alternatively, move $\hat{\theta}_k + d_k$ to the nearest valid point within Θ . Let $\theta_{\text{new}}(k+1)$ equal $\hat{\theta}_k + d_k \in \Theta$ or the aforementioned nearest valid point in Θ .

Step 2 If $L(\theta_{\text{new}}(k+1)) < L(\hat{\theta}_k)$, set $\hat{\theta}_{k+1} = \theta_{\text{new}}(k+1)$; else, set $\hat{\theta}_{k+1} = \hat{\theta}_k$.

Step 3 Stop if the maximum number of L evaluations has been reached or the user is otherwise satisfied with the current estimate for θ via appropriate stopping criteria; else, return to Step 1 with the new k set to the former $k + 1$.

For continuous problems, Matyas (1965) and others have used the (multivariate) normal distribution for generating \mathbf{d}_k . However, the user is free to set the distribution of the deviation vector \mathbf{d}_k . The distribution should have mean zero and each component should have a variation (e.g., standard deviation) consistent with the magnitudes of the corresponding $\boldsymbol{\theta}$ elements. This allows the algorithm to assign roughly equal weight to each of the components of $\boldsymbol{\theta}$ as it moves through the search space. Although not formally allowed in the convergence theory, it is often advantageous in practice if the variability in \mathbf{d}_k is reduced as k increases. This allows one to focus the search more tightly as evidence is accrued on the location of the solution (as expressed by the location of our current estimate $\hat{\boldsymbol{\theta}}_k$).

The convergence theory for the localized algorithms tends to be more restrictive than the theory for blind search. Solis and Wets (1981) provide a theorem for global convergence of localized algorithms, but the theorem conditions may not be verifiable in practice. An earlier theorem from Matyas (1965) (with proof corrected in Baba et al., 1977) provides for global convergence of the localized search above if L is a continuous function. The convergence is in the “in probability” sense. The theorem allows for more than one global minimum to exist in Θ . Therefore, in general, the result provides no guarantee of $\hat{\boldsymbol{\theta}}_k$ ever settling near any one value $\boldsymbol{\theta}^*$. We present the theorem statement below.

Convergence Theorem for Localized Search. Let Θ^* represent the set of global minima for L (see Sect. 6.1). Suppose that L is continuous on a bounded domain Θ and that if $\hat{\boldsymbol{\theta}}_k + \mathbf{d}_k \notin \Theta$ at a given iteration, a new \mathbf{d}_k is randomly generated. For any $\eta > 0$, let $R_\eta = \bigcup_{\boldsymbol{\theta}^* \in \Theta^*} \{\boldsymbol{\theta} : |L(\boldsymbol{\theta}) - L(\boldsymbol{\theta}^*)| < \eta\}$. Then, for \mathbf{d}_k having an i.i.d. $N(\mathbf{0}, \mathbf{I}_p)$ distribution, $\lim_{k \rightarrow \infty} P(\hat{\boldsymbol{\theta}}_k \in R_\eta) = 1$.

The above algorithm might be considered the most naïve of the localized random search algorithms. More sophisticated approaches are also easy to implement. For instance, if a search in one direction increases L , then it is likely to be beneficial to move in the opposite direction. Further, successive iterations in a direction that tend to consistently reduce L should encourage further iterations in the same direction. Many algorithms exploiting these simple properties exist (e.g., Solis and Wets, 1981, and Zhigljavsky, 1991).

In spite of its simplicity, the localized search algorithm is surprisingly effective in a wide range of problems. Several demonstrations are given in Sects. 2.2.3 and 2.3 in Spall (2003).

The random search algorithms above are usually based on perfect (noise-free) measurements of the loss function. This is generally considered a critical part of such algorithms (Pflug, 1996, p. 25). In contrast to the noise-free case, random search methods with noisy loss evaluations of the form $\gamma(\boldsymbol{\theta}) = L(\boldsymbol{\theta}) + \varepsilon(\boldsymbol{\theta})$ generally do not formally converge. However, there are means by which the random search techniques can be modified to accommodate noisy measurements, at least on a heuristic basis.

Some of the limited formal convergence theory for random search as applied to the noisy measurement case includes Yakowitz and Fisher (1973, Sect. 4) and

Zhigljavsky (1991, Chap. 3). Spall (2003, Sect. 2.3) discusses some practical methods for coping with noise, including simple averaging of the noisy loss function evaluations $y(\boldsymbol{\theta})$ at each value of $\boldsymbol{\theta}$ generated in the search process and a modification of the algorithm's key decision criterion (step 1 of blind random search and step 2 of localized random search) to build in some robustness to the noise. However, the averaging method can be costly since the error decreases only at the rate of $1/\sqrt{N}$ when averaging N function evaluations with independent noise. Likewise, the altered threshold may be costly by rejecting too many changes in $\boldsymbol{\theta}$ due to the conservative nature of the modified criterion. The presence of noise in the loss evaluations makes the optimization problem so much more challenging that there is little choice but to accept these penalties if one wants to use a simple random search. We will see in the next section that stochastic approximation tends to be more adept at coping with noise at the price of a more restrictive problem setting than the noise-free convergence theorem above.

6.3 Stochastic Approximation

6.3.1 Introduction

Stochastic approximation (SA) is a cornerstone of stochastic optimization. Robbins and Monro (1951) introduced SA as a general root-finding method when only noisy measurements of the underlying function are available. Let us now discuss some aspects of SA as applied to the more specific problem of root-finding in the context of optimization. With a differentiable loss function $L(\boldsymbol{\theta})$, recall the familiar set of p equations and p unknowns for use in finding a minimum $\boldsymbol{\theta}^*$:

$$\mathbf{g}(\boldsymbol{\theta}) = \frac{\partial L}{\partial \boldsymbol{\theta}} = \mathbf{0} . \quad (6.3)$$

(Of course, side conditions are required to guarantee that a root of (6.3) is a minimum, not a maximum or saddlepoint.) Note that (6.3) is nominally only directed at local optimization problems, although some extensions to global optimization are possible, as briefly discussed in Sect. 6.3.3. There are a number of approaches for solving the problem represented by (6.3) when *direct* (usually noisy) measurements of the gradient \mathbf{g} are available. These typically go by the name of *stochastic gradient* methods (e.g., Spall, 2003, Chap. 5). In contrast to the stochastic gradient approach – but consistent with the emphasis in the random search and genetic algorithms (Sects. 6.2 and 6.4 here) – let us focus on SA when only measurements of L are available. However, unlike the emphasis in random search and genetic algorithms, we consider *noisy* measurements of L .

To motivate the general SA approach, first recall the familiar form for the unconstrained deterministic steepest descent algorithm for solving (6.3):

$$\hat{\boldsymbol{\theta}}_{k+1} = \hat{\boldsymbol{\theta}}_k - a_k \mathbf{g}(\hat{\boldsymbol{\theta}}_k) ,$$

where the gain (or step size) satisfies $a_k > 0$ (see, e.g., Bazaraa et al., 1993, pp. 300–308 or any other book on mathematical programming; Spall, 2003, Sect. 1.4). This algorithm requires exact knowledge of \mathbf{g} . Steepest descent will converge to $\boldsymbol{\theta}^*$ under certain fairly general conditions. (A notable variation of steepest descent is the Newton–Raphson algorithm [sometimes called Newton’s method; e.g., Bazaraa et al., 1993, pp. 308–312], which has the form $\hat{\boldsymbol{\theta}}_{k+1} = \hat{\boldsymbol{\theta}}_k - a_k \mathbf{H}(\hat{\boldsymbol{\theta}}_k)^{-1} \mathbf{g}(\hat{\boldsymbol{\theta}}_k)$, where $\mathbf{H}(\cdot)$ is the Hessian (second derivative) matrix of L . Under more restrictive conditions, the Newton–Raphson algorithm has a much faster rate of convergence to $\boldsymbol{\theta}^*$ than steepest descent. However, with its requirement for a Hessian matrix, it is generally more challenging to implement. An SA version of Newton–Raphson is discussed briefly at the end of Sect. 6.3.3.)

Unlike with steepest descent, it is assumed here that we have no direct knowledge of \mathbf{g} . The recursive procedure of interest is in the general SA form

$$\hat{\boldsymbol{\theta}}_{k+1} = \hat{\boldsymbol{\theta}}_k - a_k \hat{\mathbf{g}}_k(\hat{\boldsymbol{\theta}}_k), \quad (6.4)$$

where $\hat{\mathbf{g}}_k(\hat{\boldsymbol{\theta}}_k)$ is the estimate of \mathbf{g} at the iterate $\hat{\boldsymbol{\theta}}_k$ based on measurements of the loss function. Hence, (6.4) is analogous to the steepest descent algorithm, with the gradient estimate $\hat{\mathbf{g}}_k(\boldsymbol{\theta})$ replacing the direct gradient \mathbf{g} at $\boldsymbol{\theta} = \hat{\boldsymbol{\theta}}_k$. The gain $a_k > 0$ here also acts in a way similar to its role in the steepest descent form. Under appropriate conditions, the iteration in (6.4) converges to $\boldsymbol{\theta}^*$ in some stochastic sense (usually almost surely, a.s.). (There are constrained forms of SA, but we do not discuss those here; see, e.g., Spall, 2003, Chaps. 4–7).

Sections 6.3.2 and 6.3.3 discuss two SA methods for carrying out the optimization task using noisy measurements of the loss function. Section 6.3.2 discusses the traditional finite-difference SA method and Sect. 6.3.3 discusses the more recent simultaneous perturbation method.

Finite-Difference SA

6.3.2

The essential part of (6.4) is the gradient approximation $\hat{\mathbf{g}}_k(\hat{\boldsymbol{\theta}}_k)$. The traditional means of forming the approximation is the finite-difference method. Expression (6.4) with this approximation represents the finite-difference SA (FDSA) algorithm. One-sided gradient approximations involve measurements $\gamma(\hat{\boldsymbol{\theta}}_k)$ and $\gamma(\hat{\boldsymbol{\theta}}_k + \text{perturbation})$, while two-sided approximations involve measurements of the form $\gamma(\hat{\boldsymbol{\theta}}_k \pm \text{perturbation})$. The two-sided FD approximation for use with (6.4) is

$$\hat{\mathbf{g}}_k(\hat{\boldsymbol{\theta}}_k) = \begin{bmatrix} \frac{\gamma(\hat{\boldsymbol{\theta}}_k + c_k \boldsymbol{\xi}_1) - \gamma(\hat{\boldsymbol{\theta}}_k - c_k \boldsymbol{\xi}_1)}{2c_k} \\ \vdots \\ \frac{\gamma(\hat{\boldsymbol{\theta}}_k + c_k \boldsymbol{\xi}_p) - \gamma(\hat{\boldsymbol{\theta}}_k - c_k \boldsymbol{\xi}_p)}{2c_k} \end{bmatrix}, \quad (6.5)$$

where $\boldsymbol{\xi}_i$ denotes a vector with a 1 in the i th place and 0’s elsewhere and $c_k > 0$ defines the difference magnitude. The pair $\{a_k, c_k\}$ are the gains (or gain sequences)

for the FDSA algorithm. The two-sided form in (6.5) is the obvious multivariate extension of the scalar two-sided form in Kiefer and Wolfowitz (1952). The initial multivariate method in Blum (1954) used a one-sided approximation.

It is of fundamental importance to determine conditions such that $\hat{\theta}_k$ as shown in (6.4) and (6.5) converges to θ^* in some appropriate stochastic sense. The convergence theory for the FDSA algorithm is similar to “standard” convergence theory for the root-finding SA algorithm of Robbins and Monro (1951). Additional difficulties, however, arise due to a bias in $\hat{g}_k(\hat{\theta}_k)$ as an estimator of $g(\hat{\theta}_k)$. That is, standard conditions for convergence of SA require unbiased estimates of $g(\cdot)$ at all k . On the other hand, $\hat{g}_k(\hat{\theta}_k)$, as shown in (6.5), is a biased estimator, with the bias having a magnitude of order c_k^2 . We will not present the details of the convergence theory here, as it is available in many other references (e.g., Fabian, 1971; Kushner and Yin, 1997, Sects. 5.3, 8.3, and 10.3; Ruppert, 1991; Spall, 2003, Chap. 6). However, let us note that the standard conditions on the gain sequences are: $a_k > 0$, $c_k > 0$, $a_k \rightarrow 0$, $c_k \rightarrow 0$, $\sum_{k=0}^{\infty} a_k = \infty$, and $\sum_{k=0}^{\infty} a_k^2/c_k^2 < \infty$. The choice of these gain sequences is critical to the performance of the method. Common forms for the sequences are:

$$a_k = \frac{a}{(k+1+A)^\alpha} \quad \text{and} \quad c_k = \frac{c}{(k+1)^\gamma},$$

where the coefficients a , c , α , and γ are strictly positive and $A \geq 0$. The user must choose these coefficients, a process usually based on a combination of the theoretical restrictions above, trial-and-error numerical experimentation, and basic problem knowledge. In some cases, it is possible to partially automate the selection of the gains (see, e.g., Spall, 2003, Sect. 6.6).

Let us summarize a numerical example based on the following $p = 10$ loss function:

$$L(\theta) = \theta^T \mathbf{B}^T \mathbf{B} \theta + 0.1 \sum_{i=1}^{10} (\mathbf{B}\theta)_i^3 + 0.01 \sum_{i=1}^{10} (\mathbf{B}\theta)_i^4,$$

where $(\cdot)_i$ represents the i th component of the argument vector $\mathbf{B}\theta$, and \mathbf{B} is such that $10\mathbf{B}$ is an upper triangular matrix of 1's. The minimum occurs at $\theta^* = \mathbf{0}$ with $L(\theta^*) = 0$; all runs are initialized at $\hat{\theta}_0 = [1, 1, \dots, 1]^T$ (so $L(\hat{\theta}_0) = 4.178$). Suppose that the measurement noise ε is independent, identically distributed (i.i.d.) $N(0, 1)$. All iterates θ_k are constrained to be in $\Theta = [-5, 5]^{10}$. If an iterate falls outside of Θ , each individual component of the candidate θ that violates the interval $[-5, 5]$ is mapped to its nearest endpoint ± 5 . The subsequent gradient estimate is formed at the modified (valid) θ value. (The perturbed values $\hat{\theta}_k \pm c_k \xi_i$ are allowed to go outside of Θ .)

Using $n = 1000$ loss measurements per run, we compare FDSA with the localized random search method of Sect. 6.2. Based on principles for gain selection in Spall (2003, Sect. 6.6) together with some limited trial-and-error experimentation, we chose $a = 0.5$, $c = 1$, $A = 5$, $\alpha = 0.602$, and $\gamma = 0.101$ for FDSA and an average of 20 loss measurements per iteration with normally distributed perturbations having distribution $N(\mathbf{0}, 0.5^2 \mathbf{I}_{10})$ for the random search method.

Figure 6.2 summarizes the results. Each curve represents the sample mean of 50 independent replications. An individual replication of one of the two algorithms has much more variation than the corresponding smoothed curve in the figure.

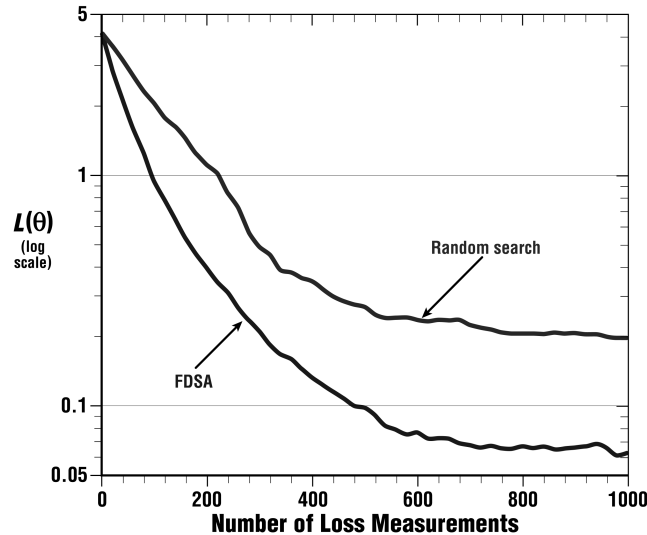


Figure 6.2. Comparison of FDSA and localized random search. Each curve represents sample mean of 50 independent replications

Figure 6.2 shows that both algorithms produce an overall reduction in the (true) loss function as the number of measurements approach 1000. The curves illustrate that FDSA outperforms random search in this case. To make the comparison fair, attempts were made to tune each algorithm to provide approximately the best performance possible. Of course, one must be careful about using this example to infer that such a result will hold in other problems as well.

Simultaneous Perturbation SA

6.3.3

The FDSA algorithm of Sect. 6.3.2 is a standard SA method for carrying out optimization with noisy measurement of the loss function. However, as the dimension p grows large, the number of loss measurements required may become prohibitive. That is, each two-sided gradient approximation requires $2p$ loss measurements. More recently, the simultaneous perturbation SA (SPSA) method was introduced, requiring only two measurements per iteration to form a gradient approximation independent of the dimension p . This provides the potential for a large savings in the overall cost of optimization.

Beginning with the generic SA form in (6.4), we now present the SP form of the gradient approximation. In this form, all elements of $\hat{\theta}_k$ are randomly perturbed

together to obtain two loss measurements $y(\cdot)$. For the two-sided SP gradient approximation, this leads to

$$\begin{aligned} \hat{\mathbf{g}}_k(\hat{\boldsymbol{\theta}}_k) &= \begin{bmatrix} \frac{y(\hat{\boldsymbol{\theta}}_k + c_k \boldsymbol{\Delta}_k) - y(\hat{\boldsymbol{\theta}}_k - c_k \boldsymbol{\Delta}_k)}{2c_k \Delta_{k1}} \\ \vdots \\ \frac{y(\hat{\boldsymbol{\theta}}_k + c_k \boldsymbol{\Delta}_k) - y(\hat{\boldsymbol{\theta}}_k - c_k \boldsymbol{\Delta}_k)}{2c_k \Delta_{kp}} \end{bmatrix} \\ &= \frac{y(\hat{\boldsymbol{\theta}}_k + c_k \boldsymbol{\Delta}_k) - y(\hat{\boldsymbol{\theta}}_k - c_k \boldsymbol{\Delta}_k)}{2c_k} \left[\Delta_{k1}^{-1}, \Delta_{k2}^{-1}, \dots, \Delta_{kp}^{-1} \right]^T, \end{aligned} \quad (6.6)$$

where the mean-zero p -dimensional random perturbation vector, $\boldsymbol{\Delta}_k = [\Delta_{k1}, \Delta_{k2}, \dots, \Delta_{kp}]^T$, has a user-specified distribution satisfying certain conditions and c_k is a positive scalar (as with FDSA). Because the numerator is the same in all p components of $\hat{\mathbf{g}}_k(\hat{\boldsymbol{\theta}}_k)$, the number of loss measurements needed to estimate the gradient in SPSA is *two*, regardless of the dimension p .

Relative to FDSA, the p -fold measurement savings per iteration, of course, provides only the *potential* for SPSA to achieve large savings in the total number of measurements required to estimate $\boldsymbol{\theta}$ when p is large. This potential is realized if the number of iterations required for effective convergence to an optimum $\boldsymbol{\theta}^*$ does not increase in a way to cancel the measurement savings per gradient approximation. One can use asymptotic distribution theory to address this issue. In particular, both FDSA and SPSA are known to be asymptotically normally distributed under very similar conditions. One can use this asymptotic distribution result to characterize the mean-squared error $E\left(\|\hat{\boldsymbol{\theta}}_k - \boldsymbol{\theta}^*\|^2\right)$ for the two algorithms for large k . Fortunately, under fairly broad conditions, the p -fold savings *at* each iteration is preserved *across* iterations. In particular, based on asymptotic considerations:

Under reasonably general conditions (see Spall, 1992, or Spall, 2003, Chap. 7), the SPSA and FDSA algorithms achieve the same level of statistical accuracy for a given number of iterations even though SPSA uses only $1/p$ times the number of function evaluations of FDSA (since each gradient approximation uses only $1/p$ the number of function evaluations).

The SPSA Web site (www.jhuapl.edu/SPSA) includes many references on the theory and application of SPSA. On this Web site, one can find many accounts of numerical studies that are consistent with the efficiency statement above. (Of course, given that the statement is based on asymptotic arguments and associated regularity conditions, one should not assume that the result always holds.) In addition, there are references describing many applications. These include queuing systems, pattern recognition, industrial quality improvement, aircraft design, simulation-based optimization, bioprocess control, neural network train-

ing, chemical process control, fault detection, human-machine interaction, sensor placement and configuration, and vehicle traffic management.

We will not present the formal conditions for convergence and asymptotic normality of SPSA, as such conditions are available in many references (e.g., Dippon and Renz, 1997; Gerencsér, 1999; Spall, 1992, 2003, Chap. 7). These conditions are essentially identical to the standard conditions for convergence of SA algorithms, with the exception of the additional conditions on the user-generated perturbation vector Δ_k .

The choice of the distribution for generating the Δ_k is important to the performance of the algorithm. The standard conditions for the elements Δ_{ki} of Δ_k are that the $\{\Delta_{ki}\}$ are independent for all k, i , identically distributed for all i at each k , symmetrically distributed about zero and uniformly bounded in magnitude for all k . In addition, there is an important inverse moments condition:

$$E\left(\left|\frac{1}{\Delta_{ki}}\right|^{2+2\tau}\right) \leq C$$

for some $\tau > 0$ and $C > 0$. The role of this condition is to control the variation of the elements of $\hat{g}_k(\theta_k)$ (which have Δ_{ki} in the denominator). One simple and popular distribution that satisfies the inverse moments condition is the symmetric Bernoulli ± 1 distribution. (In fact, as discussed in Spall, 2003, Sect. 7.7, this distribution can be shown to be optimal under general conditions when using asymptotic considerations.) Two common mean-zero distributions that do *not* satisfy the inverse moments condition are symmetric uniform and normal with mean zero. The failure of both of these distributions is a consequence of the amount of probability mass near zero. Exercise 7.3 in Spall (2003) illustrates the dramatic performance degradation that can occur through using distributions that violate the inverse moments condition.

As with any real-world implementation of stochastic optimization, there are important practical considerations when using SPSA. One is to attempt to define θ so that the magnitudes of the θ elements are similar to one another. This desire is apparent by noting that the magnitudes of all components in the perturbations $c_k \Delta_k$ are identical in the case where identical Bernoulli distributions are used. Although it is not always possible to choose the definition of the elements in θ , in most cases an analyst will have the flexibility to specify the units for θ to ensure similar magnitudes. Another important consideration is the choice of the gains a_k, c_k . The principles described for FDSA above apply to SPSA as well. Section 7.5 of Spall (2003) provides additional practical guidance.

There have been a number of important extensions of the basic SPSA method represented by the combination of (6.4) and (6.6). Three such extensions are to the problem of global (versus local) optimization, to discrete (versus continuous) problems, and to include second-order-type information (Hessian matrix) with the aim of creating a stochastic analogue to the deterministic Newton–Raphson method.

The use of SPSA for *global* minimization among multiple local minima is discussed in Maryak and Chin (2001). One of their approaches relies on injecting Monte Carlo noise in the right-hand side of the basic SPSA updating step in (6.4). This approach is a common way of converting SA algorithms to global optimizers through the additional “bounce” introduced into the algorithm (Yin, 1999). Maryak and Chin (2001) also show that basic SPSA *without* injected noise (i.e., (6.4) and (6.6) without modification) may, under certain conditions, be a global optimizer. Formal justification for this result follows because the random error in the SP gradient approximation acts in a way that is statistically equivalent to the injected noise mentioned above.

Discrete optimization problems (where θ may take on discrete or combined discrete/continuous values) are discussed in Gerencsér et al. (1999). Discrete SPSA relies on a fixed-gain (constant a_k and c_k) version of the standard SPSA method. The parameter estimates produced are constrained to lie on a discrete-valued grid. Although gradients do not exist in this setting, the approximation in (6.6) (appropriately modified) is still useful as an efficient measure of slope information.

Finally, using the simultaneous perturbation idea, it is possible to construct a simple method for estimating the Hessian (or Jacobian) matrix of L while, concurrently, estimating the primary parameters of interest (θ). This adaptive SPSA (ASP) approach produces a stochastic analogue to the deterministic Newton–Raphson algorithm (e.g., Bazaraa et al., 1993, pp. 308–312), leading to a recursion that is optimal or near-optimal in its rate of convergence and asymptotic error. The approach applies in both the gradient-free setting emphasized in this section and in the root-finding/stochastic gradient-based (Robbins–Monro) setting reviewed in Spall (2003, Chaps. 4 and 5). Like the standard SPSA algorithm, the ASP algorithm requires only a small number of loss function (or gradient, if relevant) measurements per iteration – independent of the problem dimension – to adaptively estimate the Hessian and parameters of primary interest. Further information is available at Spall (2000) or Spall (2003, Sect. 7.8).

6.4 Genetic Algorithms

6.4.1 Introduction

Genetic algorithms (GAs) represent a popular approach to stochastic optimization, especially as relates to the global optimization problem of finding the best solution among multiple local minima. (GAs may be used in general search problems that are not directly represented as stochastic optimization problems, but we focus here on their use in optimization.) GAs represent a special case of the more general class of evolutionary computation algorithms (which also includes methods such as evolutionary programming and evolution strategies). The GA applies when the elements of θ are real-, discrete-, or complex-valued. As suggested by the name, the GA is based loosely on principles of natural evolution and survival of the fittest.

In fact, in GA terminology, an equivalent *maximization* criterion, such as $-L(\theta)$ (or its analogue based on a bit-string form of θ), is often referred to as the *fitness function* to emphasize the evolutionary concept of the fittest of a species.

A fundamental difference between GAs and the random search and SA algorithms considered in Sects. 6.2 and 6.3 is that GAs work with a *population* of candidate solutions to the problem. The previous algorithms worked with one solution and moved toward the optimum by updating this one estimate. GAs simultaneously consider multiple candidate solutions to the problem of minimizing L and iterate by moving this population of solutions toward a global optimum. The terms *generation* and *iteration* are used interchangeably to describe the process of transforming one population of solutions to another. Figure 6.3 illustrates the successful operations of a GA for a population of size 12 with problem dimension $p = 2$. In this conceptual illustration, the population of solutions eventually come together at the global optimum.

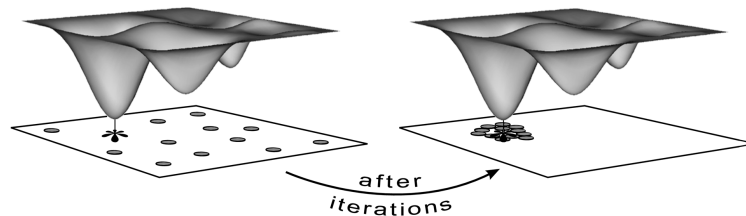


Figure 6.3. Minimization of multimodal loss function. Successful operations of a GA with a population of 12 candidate solutions clustering around the global minimum after some number of iterations (generations) (Reprinted from *Introduction to Stochastic Search and Optimization* with permission of John Wiley & Sons, Inc.)

The use of a population versus a single solution affects in a basic way the range of practical problems that can be considered. In particular, GAs tend to be best suited to problems where the loss function evaluations are computer-based calculations such as complex function evaluations or simulations. This contrasts with the single-solution approaches discussed earlier, where the loss function evaluations may represent computer-based calculations *or* physical experiments. Population-based approaches are not generally feasible when working with real-time physical experiments. Implementing a GA with physical experiments requires that either there be multiple identical experimental setups (parallel processing) or that the single experimental apparatus be set to the same state prior to each population member's loss evaluation (serial processing). These situations do not occur often in practice.

Specific values of θ in the population are referred to as *chromosomes*. The central idea in a GA is to move a set (population) of chromosomes from an initial collection of values to a point where the fitness function is optimized. We let N denote the population size (number of chromosomes in the population). Most of the early work in the field came from those in the fields of computer science and artificial intelligence. More recently, interest has extended to essentially all

branches of business, engineering, and science where search and optimization are of interest. The widespread interest in GAs appears to be due to the success in solving many difficult optimization problems. Unfortunately, to an extent greater than with other methods, some interest appears also to be due to a regrettable amount of “salesmanship” and exaggerated claims. (For example, in a recent software advertisement, the claim is made that the software “... uses GAs to solve *any* optimization problem.” Such statements are provably false.) While GAs are important tools within stochastic optimization, there is no formal evidence of consistently superior performance – relative to other appropriate types of stochastic algorithms – in any broad, identifiable class of problems.

Let us now give a very brief historical account. The reader is directed to Goldberg (1989, Chap. 4), Mitchell (1996, Chap. 1), Michalewicz (1996, pp. 1–10), Fogel (2000, Chap. 3), and Spall (2003, Sect. 9.2) for more complete historical discussions. There had been some success in creating mathematical analogues of biological evolution for purposes of search and optimization since at least the 1950s (e.g., Box, 1957). The cornerstones of modern evolutionary computation – evolution strategies, evolutionary programming, and GAs – were developed independently of each other in the 1960s and 1970s. John Holland at the University of Michigan published the seminal monograph *Adaptation in Natural and Artificial Systems* (Holland, 1975). There was subsequently a sprinkle of publications, leading to the first full-fledged textbook Goldberg (1989). Activity in GAs grew rapidly beginning in the mid-1980s, roughly coinciding with resurgent activity in other artificial intelligence-type areas such as neural networks and fuzzy logic. There are now many conferences and books in the area of evolutionary computation (especially GAs), together with countless other publications.

6.4.2 Chromosome Coding and the Basic GA Operations

This section summarizes some aspects of the encoding process for the population chromosomes and discusses the *selection*, *elitism*, *crossover*, and *mutation* operations. These operations are combined to produce the steps of the GA.

An essential aspect of GAs is the encoding of the N values of θ appearing in the population. This encoding is critical to the GA operations and the associated decoding to return to the natural problem space in θ . Standard binary (0, 1) bit strings have traditionally been the most common encoding method, but other methods include gray coding (which also uses (0, 1) strings, but differs in the way the bits are arranged) and basic computer-based floating-point representation of the real numbers in θ . This 10-character coding is often referred to as *real-number coding* since it operates as if working with θ directly. Based largely on successful numerical implementations, this natural representation of θ has grown more popular over time. Details and further references on the above and other coding schemes are given in Michalewicz (1996, Chap. 5), Mitchell (1996, Sects. 5.2 and 5.3), Fogel (2000, Sects. 3.5 and 4.3), and Spall (2003, Sect. 9.3).

Let us now describe the basic operations mentioned above. For consistency with standard GA terminology, let us assume that $L(\theta)$ has been transformed

to a fitness function with higher values being better. A common transformation is to simply set the fitness function to $-L(\theta) + C$, where $C \geq 0$ is a constant that ensures that the fitness function is nonnegative on Θ (nonnegativity is only required in some GA implementations). Hence, the operations below are described for a *maximization* problem. It is also assumed here that the fitness evaluations are noise-free. Unless otherwise noted, the operations below apply with any coding scheme for the chromosomes.

selection and *elitism* steps occur after evaluating the fitness function for the current population of chromosomes. A subset of chromosomes is selected to use as parents for the succeeding generation. This operation is where the survival of the fittest principle arises, as the parents are chosen according to their fitness value. While the aim is to emphasize the fitter chromosomes in the selection process, it is important that not *too* much priority is given to the chromosomes with the highest fitness values early in the optimization process. Too much emphasis of the fitter chromosomes may tend to reduce the diversity needed for an adequate search of the domain of interest, possibly causing premature convergence in a local optimum. Hence methods for selection allow with some nonzero probability the selection of chromosomes that are suboptimal.

Associated with the selection step is the optional “elitism” strategy, where the $N_e < N$ best chromosomes (as determined from their fitness evaluations) are placed directly into the next generation. This guarantees the preservation of the N_e best chromosomes at each generation. Note that the elitist chromosomes in the original population are also eligible for selection and subsequent recombination.

As with the coding operation for θ , many schemes have been proposed for the selection process of choosing parents for subsequent recombination. One of the most popular methods is *roulette wheel selection* (also called *fitness proportionate selection*). In this selection method, the fitness functions must be nonnegative on Θ . An individual’s slice of a Monte Carlo-based roulette wheel is an area proportional to its fitness. The “wheel” is spun in a simulated fashion $N - N_e$ times and the parents are chosen based on where the pointer stops. Another popular approach is called *tournament selection*. In this method, chromosomes are compared in a “tournament,” with the better chromosome being more likely to win. The tournament process is continued by sampling (with replacement) from the original population until a full complement of parents has been chosen. The most common tournament method is the binary approach, where one selects two pairs of chromosomes and chooses as the two parents the chromosome in each pair having the higher fitness value. Empirical evidence suggests that the tournament selection method often performs better than roulette selection. (Unlike tournament selection, roulette selection is very sensitive to the scaling of the fitness function.) Mitchell (1996, Sect. 5.4) provides a good survey of several other selection methods.

The *crossover* operation creates offspring of the pairs of parents from the selection step. A crossover probability P_c is used to determine if the offspring will represent a blend of the chromosomes of the parents. If no crossover takes place, then the two offspring are clones of the two parents. If crossover does take

place, then the two offspring are produced according to an interchange of parts of the chromosome structure of the two parents. Figure 6.4 illustrates this for the case of a ten-bit binary representation of the chromosomes. This example shows one-point crossover, where the bits appearing after one randomly chosen dividing (splice) point in the chromosome are interchanged. In general, one can have a number of splice points up to the number of bits in the chromosomes minus one, but one-point crossover appears to be the most commonly used.

Note that the crossover operator also applies directly with real-number coding since there is nothing directly connected to binary coding in crossover. All that is required are two lists of compatible symbols. For example, one-point crossover applied to the chromosomes (θ values) $[6.7, -7.4, 4.0, 3.9|6.2, -1.5]$ and $[-3.8, 5.3, 9.2, -0.6|8.4, -5.1]$ yields the two children: $[6.7, -7.4, 4.0, 3.9, 8.4, -5.1]$ and $[-3.8, 5.3, 9.2, -0.6, 6.2, -1.5]$.

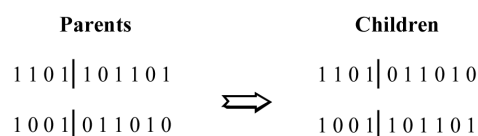


Figure 6.4. Example of crossover operator under binary coding with one splice point

The final operation we discuss is *mutation*. Because the initial population may not contain enough variability to find the solution via crossover operations alone, the GA also uses a mutation operator where the chromosomes are randomly changed. For the binary coding, the mutation is usually done on a bit-by-bit basis where a chosen bit is flipped from 0 to 1, or vice versa. Mutation of a given bit occurs with small probability P_m . Real-number coding requires a different type of mutation operator. That is, with a (0, 1)-based coding, an opposite is uniquely defined, but with a real number, there is no clearly defined opposite (e.g., it does not make sense to “flip” the 2.74 element). Probably the most common type of mutation operator is simply to add small independent normal (or other) random vectors to each of the chromosomes (the θ values) in the population.

As discussed in Sect. 6.1.4, there is no easy way to know when a stochastic optimization algorithm has effectively converged to an optimum. This includes genetic algorithms. The one obvious means of stopping a GA is to end the search when a budget of fitness (equivalently, loss) function evaluations has been spent. Alternatively, termination may be performed heuristically based on subjective and objective impressions about convergence. In the case where noise-free fitness measurements are available, criteria based on fitness evaluations may be most useful. For example, a fairly natural criterion suggested in Schwefel (1995, p. 145) is to stop when the maximum and minimum fitness values over the N population values within a generation are sufficiently close to one another. However, this criterion provides no formal guarantee that the algorithm has found a global solution.

The Core Genetic Algorithm

6.4.3

The steps of a basic form of the GA are given below. These steps are general enough to govern many (perhaps most) modern implementations of GAs, including those in modern commercial software. Of course, the performance of a GA typically depends greatly on the implementation details, just as with other stochastic optimization algorithms. Some of these practical implementation issues are taken up in the next section.

Core GA Steps for Noise-Free Fitness Evaluations

- Step 0 (Initialization)* Randomly generate an initial population of N chromosomes and evaluate the fitness function (the conversion of $L(\theta)$ to a function to be maximized for the encoded version of θ) for each of the chromosomes.
- Step 1 (Parent Selection)* Set $N_e = 0$ if elitism strategy is not used; $0 < N_e < N$ otherwise. Select with replacement $N - N_e$ parents from the full population (including the N_e elitist elements). The parents are selected according to their fitness, with those chromosomes having a higher fitness value being selected more often.
- Step 2 (Crossover)* For each pair of parents identified in Step 1, perform crossover on the parents at a randomly (perhaps uniformly) chosen splice point (or *points* if using multi-point crossover) with probability P_c . If no crossover takes place (probability $1 - P_c$), then form two offspring that are exact copies (clones) of the two parents.
- Step 3 (Replacement and Mutation)* While retaining the N_e best chromosomes from the previous generation, replace the remaining $N - N_e$ chromosomes with the current population of offspring from Step 2. For the bit-based implementations, mutate the individual bits with probability P_m ; for real coded implementations, use an alternative form of “small” modification (in either case, one has the option of choosing whether to make the N_e elitist chromosomes candidates for mutation).
- Step 4 (Fitness and End Test)* Compute the fitness values for the new population of N chromosomes. Terminate the algorithm if the stopping criterion is met or if the budget of fitness function evaluations is exhausted; else return to Step 1.

Some Implementation Aspects

6.4.4

While the above steps provide the broad outline for many modern implementations of GAs, there are some important implementation aspects that must be decided before a practical implementation. This section outlines a few of those aspects. More detailed discussions are given in Mitchell (1996, Chap. 5), Michalewicz (1996, Chaps. 4–6), Fogel (2000, Chaps. 3 and 4), Goldberg (2002, Chap. 12), and other references mentioned below. A countless number of numerical studies have been reported in the literature; we do not add to that list here.

As with other stochastic optimization methods, the choice of algorithm-specific coefficients has a significant impact on performance. With GAs, there is a relatively large number of user decisions required. The following must be set: the choice of chromosome encoding, the population size (N), the probability distribution generating the initial population, the strategy for parent selection (roulette wheel or otherwise), the number of splice points in the crossover, the crossover probability (P_c), the mutation probability (P_m), the number of retained chromosomes in elitism (N_e), and some termination criterion. Some typical values for these quantities are discussed, for example, in Mitchell (1996, pp. 175–177) and Spall (2003, Sect. 9.6).

Constraints on $L(\theta)$ (or the equivalent fitness function) and/or θ are of major importance in practice. The bit-based implementation of GAs provide a natural way of implementing component-wise lower and upper bounds on the elements of θ (i.e., a hypercube constraint). More general approaches to handling constraints are discussed in Michalewicz (1996, Chap. 7 and Sects. 4.5 and 15.3) and Michalewicz and Fogel (2000, Chap. 9).

Until now, it has been assumed that the fitness function is observed without noise. One of the two possible defining characteristics of stochastic optimization, however, is optimization with noise in the function measurements (Property I in Sect. 6.1.3). There appears to be relatively little formal analysis of GAs in the presence of noise, although the application and testing of GAs in such cases has been carried out since at least the mid-1970s (e.g., De Jong, 1975, p. 203). A large number of numerical studies are in the literature (e.g., the references and studies in Spall, 2003, Sects. 9.6 and 9.7). As with other algorithms, there is a fundamental tradeoff of more accurate information for each function input (typically, via an averaging of the inputs) and fewer function inputs versus less accurate (“raw”) information to the algorithm together with a greater number of inputs to the algorithm. There appears to be no rigorous comparison of GAs with other algorithms regarding relative robustness to noise. Regarding noise, Michalewicz and Fogel (2000, p. 325) state: “There really are no effective heuristics to guide the choices to be made that will work in general.”

6.4.5 Some Comments on the Theory for GAs

One of the key innovations in Holland (1975) was the attempt to put GAs on a stronger theoretical footing than the previous ad hoc treatments. He did this by the introduction of schema theory. While many aspects and implications of schema theory have subsequently been challenged (Reeves and Rowe, 2003, Chap. 3; Spall, 2003, Sect. 10.3), some aspects remain viable. In particular, schema theory *itself* is generally correct (subject to a few modifications), although many of the assumed *implications* have not been correct. With the appropriate caveats and restrictions, schema theory provides some intuitive explanation for the good performance that is frequently observed with GAs.

More recently, Markov chains have been used to provide a formal structure for analyzing GAs. First, let us mention one negative result. Markov chains can be used

to show that a canonical GA *without elitism* is (in general) provably nonconvergent (Rudolph, 1994). That is, with a GA that does not hold onto the best solution at each generation, there is the possibility (through crossover and mutation) that a chromosome corresponding to θ^* will be lost. (Note that the GA without elitism corresponds to the form in Holland, 1975.)

On the other hand, conditions for the formal convergence of GAs to an optimal θ^* (or its coded equivalent) are presented in Vose (1999, Chaps. 13 and 14), Fogel (2000, Chap. 4), Reeves and Rowe (2003, Chap. 6), and Spall (2003, Sect. 10.5), among other references. Consider a binary bit-coded GA with a population size of N and a string length of B bits per chromosome. Then the total number of possible *unique* populations is:

$$N_p \equiv \binom{N + 2^B - 1}{N} = \frac{(N + 2^B - 1)!}{(2^B - 1)!N!}$$

(Suzuki, 1995). It is possible to construct an $N_p \times N_p$ Markov transition matrix \mathbf{P} , where the ij th element is the probability of transitioning from the i th population of N chromosomes to the j th population of the same size. These elements depend in a nontrivial way on N , the crossover rate, and the mutation rate; the number of elite chromosomes is assumed to be $N_e = 1$ (Suzuki, 1995). Let \mathbf{p}_k be an $N_p \times 1$ vector having j th component $p_k(j)$ equal to the probability that the k th generation will result in population j , $j = 1, 2, \dots, N_p$. From basic Markov chain theory,

$$\mathbf{p}_{k+1}^T = \mathbf{p}_k^T \mathbf{P} = \mathbf{p}_0^T \mathbf{P}^{k+1},$$

where \mathbf{p}_0 is an initial probability distribution. If the chain is irreducible and ergodic (see, e.g., Spall, 2003, Appendix E), the limiting distribution of the GA (i.e., $\tilde{\mathbf{p}}^T = \lim_{k \rightarrow \infty} \mathbf{p}_k^T = \lim_{k \rightarrow \infty} \mathbf{p}_0^T \mathbf{P}^k$) exists and satisfies the stationarity equation $\tilde{\mathbf{p}}^T = \tilde{\mathbf{p}}^T \mathbf{P}$. (Recall from basic Markov chain theory that irreducibility indicates that any state may be reached from any other state after a finite number of steps.)

Suppose that θ^* is unique (i.e., Θ^* is the singleton θ^*). Let $J \subseteq \{1, 2, \dots, N_p\}$ be the set of indices corresponding to the populations that contain at least one chromosome representing θ^* . So, for example, if $J = \{1, 6, N_p - 3\}$, then each of the three populations indexed by 1, 6 and $N_p - 3$ contains at least one chromosome that, when decoded, is equal to θ^* . Under the above-mentioned assumptions of irreducibility and ergodicity, $\sum_{i \in J} \tilde{p}_i = 1$, where \tilde{p}_i is the i th element of $\tilde{\mathbf{p}}$. Hence, a GA with $N_e = 1$ and a transition matrix that is irreducible and ergodic converges in probability to θ^* .

To establish the fact of convergence alone, it may not be necessary to compute the \mathbf{P} matrix. Rather, it suffices to know that the chain is irreducible and ergodic. (For example, Rudolph, 1997, p. 125, shows that the Markov chain approach yields convergence when $0 < P_m < 1$.) However, \mathbf{P} must be explicitly computed to get the *rate* of convergence information that is available from \mathbf{p}_k . This is rarely possible in practice because the number of states in the Markov chain (and hence dimension of the Markov transition matrix) grows *very* rapidly with increases in the population size

and/or the number of bits used in coding for the population elements. For example, in even a trivial problem of $N = B = 6$, there are $\sim 10^8$ states and $\sim 10^{16}$ elements in the transition matrix; this problem is much smaller than any practical GA, which can easily have 50 to 100 population elements and 15 to 40 bits per population element (leading to well over 10^{100} states, with each element in the corresponding row and column in the transition matrix requiring significant computation).

6.5 Concluding Remarks

Stochastic optimization is a major branch of computational statistics. This paper has been a whirlwind tour through some important issues and methods in stochastic optimization. Stochastic optimization applies when there are noisy measurements of the criterion being optimized and/or there is an injected Monte Carlo randomness as part of the algorithm. Of necessity, we cover only a small fraction of available methods in this relatively brief review, although the methods described (random search, stochastic approximation, and genetic algorithms) are representative of a broad range of important and widely used algorithms. Further, the treatment here on the specific algorithms is relatively brief. In particular, the subjects covered in this paper of approximately 30 pages are treated in over 160 pages in Spall (2003, Chaps. 1–2, 6–7, and 9–10) and are given an even more detailed treatment in the many specialized books or other references.

There are many challenges to carrying out real-world optimization, including the presence of noise in the function evaluations, the difficulties in distinguishing a globally optimal solution from locally optimal solutions, the “curse of dimensionality,” the difficulties associated with nontrivial constraints, and the lack of stationarity in the solution as a result of the conditions of the problem changing over time. Stochastic optimization methods are especially useful in treating some of these challenges. In particular, by definition, they are designed for noisy function evaluations. Further, when considering injected (Monte Carlo) randomness (property II in Sect. 6.1.3), certain stochastic optimization algorithms will (under conditions, of course) serve as global optimizers. That is, the injected randomness provides enough “bounce” to the algorithm to allow for escape from local minima en route to achieving a global minimum.

In summary, while classical deterministic optimization methods (linear and nonlinear programming) are effective for a range of problems, stochastic methods are able to handle many of the problems for which deterministic methods are inappropriate. It is hoped that this summary gives the reader a flavor of the issues, algorithms, and challenges in carrying out optimization in the face of stochastic effects.

Acknowledgements. I appreciate the helpful comments of Dr. Stacy Hill on a draft version of this paper. Funding was provided by the U. S. Navy (contract N00024-98-D-8124) and the JHU/APL Independent Research and Development (IRAD)

Program. Selected parts of this article have been reprinted, by permission, from J.C. Spall, *Introduction to Stochastic Search and Optimization*, ©2003 by John Wiley and Sons, Inc.

References

- Arsham, H. (1998), "Techniques for Monte Carlo Optimizing," *Monte Carlo Methods and Applications*, vol. 4, pp. 181–229.
- Baba, N., Shoman, T., and Sawaragi, Y. (1977), "A Modified Convergence Theorem for a Random Optimization Method," *Information Sciences*, vol. 13, pp. 159–166.
- Bazaraa, M.S., Sherali, H.D., and Shetty, C.M. (1993), *Nonlinear Programming: Theory and Algorithms* (2nd ed.), Wiley, New York.
- Blum, J.R. (1954), "Multidimensional Stochastic Approximation Methods," *Annals of Mathematical Statistics*, vol. 25, pp. 737–744.
- Box, G.E.P. (1957), "Evolutionary Operation: A Method for Increasing Industrial Productivity," *Journal of the Royal Statistical Society, Ser. C.*, vol. 6, pp. 81–101.
- De Jong, K.A. (1975), "An Analysis of the Behavior of a Class of Genetic Adaptive Systems," Ph.D. dissertation, University of Michigan, Ann Arbor, MI (University Microfilms no. 76–9381).
- Dippon, J. and Renz, J. (1997), "Weighted Means in Stochastic Approximation of Minima," *SIAM Journal of Control and Optimization*, vol. 35, pp. 1811–1827.
- Fabian, V. (1971), "Stochastic Approximation," in *Optimizing Methods in Statistics* (J.S. Rustigi, ed.), Academic Press, New York, pp. 439–470.
- Fogel, D.B. (2000), *Evolutionary Computation: Toward a New Philosophy of Machine Intelligence* (2nd ed.), IEEE Press, Piscataway, NJ.
- Fouskakis, D. and Draper, D. (2002), "Stochastic Optimization: A Review," *International Statistical Review*, vol. 70, pp. 315–349.
- Fu, M.C. (2002), "Optimization for Simulation: Theory vs. Practice" (with discussion by S. Andradóttir, P. Glynn, and J.P. Kelly), *INFORMS Journal on Computing*, vol. 14, pp. 192–227.
- Gentle, J.E. (2003), *Random Number Generation and Monte Carlo Methods* (2nd ed.), Springer-Verlag, New York.
- Gerencsér, L. (1999), "Convergence Rate of Moments in Stochastic Approximation with Simultaneous Perturbation Gradient Approximation and Resetting," *IEEE Transactions on Automatic Control*, vol. 44, pp. 894–905.
- Gerencsér, L., Hill, S.D., and Vágó, Z. (1999), "Fixed Gain SPSA for Discrete Optimization," in *Proceedings of the IEEE Conference on Decision and Control*, 7–10 December 1999, Phoenix, AZ, pp. 1791–1795.
- Goldberg, D.E. (1989), *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley, Reading, MA.
- Gosavi, A. (2003), *Simulation-Based Optimization: Parametric Optimization Techniques and Reinforcement Learning*, Kluwer Academic, Boston.
- Holland, J.H. (1975), *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Ann Arbor, MI.

- Karnopp, D.C. (1963), "Random Search Techniques for Optimization Problems," *Automatica*, vol. 1, pp. 111–121.
- Kiefer, J. and Wolfowitz, J. (1952), "Stochastic Estimation of a Regression Function," *Annals of Mathematical Statistics*, vol. 23, pp. 462–466.
- Kolda, T.G., Lewis, R.M., and Torczon, V. (2003), "Optimization by Direct Search: New Perspectives on Some Classical and Modern Methods," *SIAM Review*, vol. 45, pp. 385–482.
- Kushner, H.J. and Yin, G.G. (1997), *Stochastic Approximation Algorithms and Applications*, Springer-Verlag, New York.
- Maryak, J.L. and Chin, D.C. (2001), "Global Random Optimization by Simultaneous Perturbation Stochastic Approximation," in *Proceedings of the American Control Conference*, 25–27 June 2001, Arlington, VA, pp. 756–762.
- Matyas, J. (1965), "Random Optimization," *Automation and Remote Control*, vol. 26, pp. 244–251.
- Michalewicz, Z. (1996), *Genetic Algorithms + Data Structures = Evolution Programs* (3rd ed.), Springer-Verlag, New York.
- Michalewicz, Z. and Fogel, D.B. (2000), *How to Solve It: Modern Heuristics*, Springer-Verlag, New York.
- Mitchell, M. (1996), *An Introduction to Genetic Algorithms*, MIT Press, Cambridge, MA.
- Nelder, J.A. and Mead, R. (1965), "A Simplex Method for Function Minimization," *The Computer Journal*, vol. 7, pp. 308–313.
- Pflug, G.Ch. (1996), *Optimization of Stochastic Models: The Interface Between Simulation and Optimization*, Kluwer Academic, Boston.
- Reeves, C.R. and Rowe, J.E. (2003), *Genetic Algorithms – Principles and Perspectives: A Guide to GA Theory*, Kluwer Academic, Boston.
- Robbins, H. and Monroe, S. (1951), "A Stochastic Approximation Method," *Annals of Mathematical Statistics*, vol. 22, pp. 400–407.
- Rudolph, G. (1994), "Convergence Analysis of Canonical Genetic Algorithms," *IEEE Transactions on Neural Networks*, vol. 5, pp. 96–101.
- Rudolph, G. (1997), *Convergence Properties of Evolutionary Algorithms*, Verlag Kovac, Hamburg.
- Ruppert, D. (1991), "Stochastic Approximation," in *Handbook of Sequential Analysis* (B.K. Ghosh and P.K. Sen, eds.), Marcel Dekker, New York, pp. 503–529.
- Schwefel, H.-P. (1995), *Evolution and Optimum Seeking*, Wiley, New York.
- Solis, F.J. and Wets, J.B. (1981), "Minimization by Random Search Techniques," *Mathematics of Operations Research*, vol. 6, pp. 19–30.
- Spall, J.C. (1992), "Multivariate Stochastic Approximation Using a Simultaneous Perturbation Gradient Approximation," *IEEE Transactions on Automatic Control*, vol. 37, pp. 332–341.
- Spall, J.C. (2000), "Adaptive Stochastic Approximation by the Simultaneous Perturbation Method," *IEEE Transactions on Automatic Control*, vol. 45, pp. 1839–1853.
- Spall, J.C. (2003), *Introduction to Stochastic Search and Optimization: Estimation, Simulation, and Control*, Wiley, Hoboken, NJ.

- Suzuki, J. (1995), "A Markov Chain Analysis on Simple Genetic Algorithms," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 25, pp. 655–659.
- Wolpert, D.H. and Macready, W.G. (1997), "No Free Lunch Theorems for Optimization," *IEEE Transactions on Evolutionary Computation*, vol. 1, pp. 67–82.
- Yin, G. (1999), "Rates of Convergence for a Class of Global Stochastic Optimization Algorithms," *SIAM Journal on Optimization*, vol. 10, pp. 99–120.
- Zhigljavsky, A.A. (1991), *Theory of Global Random Search*, Kluwer Academic, Boston.

Transforms in Statistics

II.7

Brani Vidakovic

7.1	<i>Introduction</i>	200
7.2	<i>Fourier and Related Transforms</i>	205
	Discrete Fourier Transform	206
	Windowed Fourier Transform	208
	Hilbert Transform	209
	Wigner–Ville Transforms	210
7.3	<i>Wavelets and Other Multiscale Transforms</i>	212
	A Case Study	213
	Continuous Wavelet Transform	214
	Multiresolution Analysis	217
	Haar Wavelets	223
	Daubechies’ Wavelets	224
7.4	<i>Discrete Wavelet Transforms</i>	226
	The Cascade Algorithm	229
	Matlab Implementation of Cascade Algorithm	233
7.5	<i>Conclusion</i>	234

It is not an overstatement to say that statistics is based on various transformations of data. Basic statistical summaries such as the sample mean, variance, z-scores, histograms, etc., are all transformed data. Some more advanced summaries, such as principal components, periodograms, empirical characteristic functions, etc., are also examples of transformed data. To give a just coverage of transforms utilized in statistics will take a size of a monograph. In this chapter we will focus only on several important transforms with the emphasis on novel multiscale transforms (wavelet transforms and its relatives).

Transformations in statistics are utilized for several reasons, but unifying arguments are that transformed data

- (i) are easier to report, store, and analyze,
- (ii) comply better with a particular modeling framework, and
- (iii) allow for an additional insight to the phenomenon not available in the domain of non-transformed data.

For example, variance stabilizing transformations, symmetrizing transformations, transformations to additivity, Laplace, Fourier, Wavelet, Gabor, Wigner-Ville, Hugh, Mellin, transforms all satisfy one or more of points listed in (i–iii).

We emphasize that words *transformation* and *transform* are often used interchangeably. However, the semantic meaning of the two words seem to be slightly different. For the word *transformation*, the synonyms are alteration, evolution, change, reconfiguration. On the other hand, the word *transform* carries the meaning of a more radical change in which the nature and/or structure of the transformed object are altered. In our context, it is natural that processes which alter the data leaving them unreduced in the same domain should be called transformations (for example Box–Cox transformation) and the processes that radically change the nature, structure, domain, and dimension of data should be called transforms (for example Wigner–Ville transform).

In this chapter we focus mainly on transforms providing an additional insight on data. After the introduction discussing three examples, several important transforms are overviewed. We selected discrete Fourier, Hilbert, and Wigner–Ville transforms, discussed in Sect. 2, and given their recent popularity, continuous and discrete wavelet transforms discussed in Sects. 3 and 4.

7.1

Introduction

As an “appetizer” we give two simple examples of use of transformations in statistics, Fisher z and Box–Cox transformations as well as the empirical Fourier–Stieltjes transform.

1

Example 1 Assume that we are looking for variance transformation $Y = \vartheta(X)$, in the case where $\text{Var } X = \sigma_X^2(\mu)$ is a function of the mean $\mu = EX$. The first order Taylor expansion of $\vartheta(X)$ about mean μ is

$$\vartheta(X) = \vartheta(\mu) + (X - \mu)\vartheta'(\mu) + O[(X - \mu)^2] .$$

Ignoring quadratic and higher order terms we see that

$$E \vartheta(X) \approx 0, \quad \text{Var } \vartheta(X) \approx E [(X - \mu)^2 \vartheta'(\mu)] = [\vartheta'(x)]^2 \sigma_X^2(\mu).$$

If $\text{Var}(\vartheta(X))$ is to be c^2 , we obtain

$$[\vartheta'(x)]^2 \sigma_X^2(\mu) = c^2$$

resulting in

$$\vartheta(x) = c \int \frac{dx}{\sigma_X(x)} dx.$$

This is a theoretical basis for the so-called Fisher z -transformation.

Let $(X_{11}, X_{21}), \dots, (X_{1n}, X_{2n})$ be a sample from bivariate normal distribution $N_2(\mu_1, \mu_2, \sigma_1^2, \sigma_2^2, \rho)$, and $\bar{X}_i = 1/n \sum_{j=1}^n X_{ij}$, $i = 1, 2$.

The Pearson coefficient of linear correlation

$$r = \frac{\sum_{i=1}^n (X_{1i} - \bar{X}_1)(X_{2i} - \bar{X}_2)}{\left[\sum_{i=1}^n (X_{1i} - \bar{X}_1)^2 \cdot \sum_{i=1}^n (X_{2i} - \bar{X}_2)^2 \right]^{1/2}}$$

has a complicated distribution involving special functions, e.g., Anderson (1984, p. 113). However, it is well known that the asymptotic distribution for r is normal $N(\rho, \frac{(1-\rho^2)^2}{n})$. Since the variance is a function of mean,

$$\begin{aligned} \vartheta(\rho) &= \int \frac{c\sqrt{n}}{1-\rho^2} d\rho \\ &= \frac{c\sqrt{n}}{2} \int \left(\frac{1}{1-\rho} + \frac{1}{1+\rho} \right) d\rho \\ &= \frac{c\sqrt{n}}{2} \log \left(\frac{1+\rho}{1-\rho} \right) + k \end{aligned}$$

is known as Fisher z -transformation for the correlation coefficient (usually for $c = 1/\sqrt{n}$ and $k = 0$). Assume that r and ρ are mapped to z and ζ as

$$z = \frac{1}{2} \log \left(\frac{1+r}{1-r} \right) = \text{arctanh } r, \quad \zeta = \frac{1}{2} \log \left(\frac{1+\rho}{1-\rho} \right) = \text{arctanh } \rho.$$

The distribution of z is approximately normal $N(\zeta, 1/(n-3))$ and this approximation is quite accurate when ρ^2/n^2 is small and when n is as low as 20. The use of Fisher z -transformation is illustrated on finding the confidence intervals for ρ and testing hypotheses about ρ .

To exemplify the above, we generated $n = 30$ pairs of normally distributed random samples with theoretical correlation $\sqrt{2}/2$. This was done by generating two i.i.d. normal samples a , and b of length 30 and taking the transformation $x_1 = a + b$, $x_2 = b$. The sample correlation coefficient r is found. This was repeated

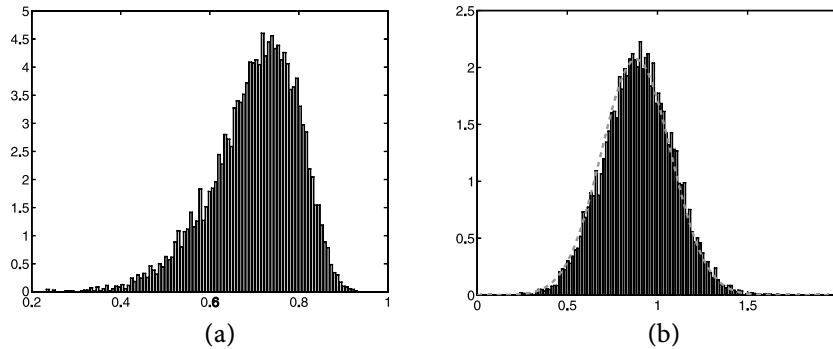


Figure 7.1. (a) Simulational run of 10,000 r 's from the bivariate population having theoretical $\rho = \sqrt{2}/2$; (b) The same r 's transformed to z 's with the normal approximation superimposed

$M = 10,000$ times. The histogram of 10,000 sample correlation coefficients is shown in Fig. 7.1a. The histogram of z -transformed r 's is shown in Fig. 7.1b with superimposed normal approximation $N(\text{arctanh}(\sqrt{2}/2), 1/(30 - 3))$.

(i) For example, $(1 - \alpha)100\%$ confidence interval for ρ is:

$$\left[\tanh\left(z - \frac{\Phi^{-1}(1 - \alpha/2)}{\sqrt{n - 3}}\right), \tanh\left(z + \frac{\Phi^{-1}(1 - \alpha/2)}{\sqrt{n - 3}}\right) \right],$$

where $z = \text{arctanh}(r)$ and $\tanh x = (e^x - e^{-x})/(e^x + e^{-x})$ and Φ stands for the standard normal cumulative distribution function.

If $r = -0.5687$ and $n = 28$ $z = -0.6456$, $z_L = -0.6456 - 1.96/5 = -1.0376$ and $z_U = -0.6456 + 1.96/5 = -0.2536$. In terms of ρ the 95% confidence interval is $[-0.7769, -0.2483]$.

(ii) Assume that two samples of size n_1 and n_2 , respectively, are obtained from two different bivariate normal populations. We are interested in testing $H_0 : \rho_1 = \rho_2$ against the two sided alternative. After observing r_1 and r_2 and transforming them to z_1 and z_2 , we conclude that the p -value of the test is $2\Phi(-|z_1 - z_2|/\sqrt{1/(n_1 - 3) + 1/(n_2 - 3)})$.

2

Example 2 Box and Cox (1964) introduced a family of transformations, indexed by real parameter λ , applicable to positive data X_1, \dots, X_n ,

$$Y_i = \begin{cases} \frac{X_i^\lambda - 1}{\lambda}, & \lambda \neq 0 \\ \log X_i, & \lambda = 0. \end{cases} \tag{7.1}$$

This transformation is mostly applied to responses in linear models exhibiting non-normality and/or heteroscedasticity. For properly selected λ , transformed

data Y_1, \dots, Y_n may look “more normal” and amenable to standard modeling techniques. The parameter λ is selected by maximizing the log-likelihood,

$$(\lambda - 1) \sum_{i=1}^n \log X_i - \frac{n}{2} \log \left[\frac{1}{n} \sum_{i=1}^n (Y_i - \bar{Y})^2 \right], \tag{7.2}$$

where Y_i are given in (7.1) and $\bar{Y} = 1/n \sum_{i=1}^n Y_i$.

As an illustration, we apply the Box–Cox transformation to apparently skewed data of CEO salaries.

Forbes magazine published data on the best small firms in 1993. These were firms with annual sales of more than five and less than \$350 million. Firms were ranked by five-year average return on investment. One of the variables extracted is the annual salary of the chief executive officer for the first 60 ranked firms (since one datum is missing, the sample size is 59). Figure 7.2a shows the histogram of row data (salaries). The data show moderate skeweness to the right. Figure 7.2b gives the values of likelihood in (7.2) for different values of λ . Note that (7.2) is maximized for λ approximately equal to 0.45. Figure 7.2c gives the transformed data by Box–Cox transformation with $\lambda = 0.45$. The histogram of transformed salaries is notably symetrized.

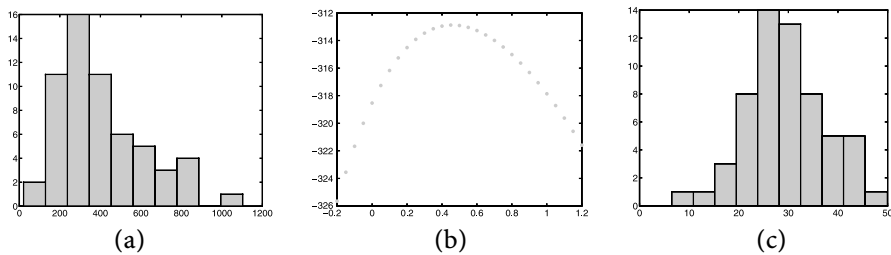


Figure 7.2. (a) Histogram of row data (CEO salaries); (b) Log-likelihood is maximized at $\lambda = 0.45$; and (c) Histogram of Box–Cox-transformed data

Example 3 As an example of transforms utilized in statistics, we provide an application of empirical Fourier–Stieltjes transform (empirical characteristic function) in testing for the independence.

3

The characteristic function of a probability distribution F is defined as its Fourier–Stieltjes transform,

$$\varphi_X(t) = E \exp(itX), \tag{7.3}$$

where E is expectation and random variable X has distribution function F . It is well known that the correspondence of characteristic functions and distribution functions is 1–1, and that closeness in the domain of characteristic functions

corresponds to closeness in the domain of distribution functions. In addition to uniqueness, characteristic functions are bounded. The same does not hold for moment generating functions which are Laplace transforms of distribution functions.

For a sample X_1, X_2, \dots, X_n one defines empirical characteristic function $\varphi^*(t)$ as

$$\varphi_X^*(t) = \frac{1}{n} \sum_{j=1}^n \exp(itX_j) .$$

The result by Feuerverger and Mureika (1977) establishes the large sample properties of the empirical characteristic function.

1

Theorem 1 For any $T < \infty$

$$P \left[\lim_{n \rightarrow \infty} \sup_{|t| \leq T} |\varphi^*(t) - \varphi(t)| = 0 \right] = 1$$

holds. Moreover, when $n \rightarrow \infty$, the stochastic process

$$Y_n(t) = \sqrt{n} (\varphi^*(t) - \varphi(t)) , \quad |t| \leq T ,$$

converges in distribution to a complex-valued Gaussian zero-mean process $Y(t)$ satisfying $Y(t) = \overline{Y(-t)}$ and

$$E \left(Y(t) \overline{Y(s)} \right) = \varphi(t+s) - \varphi(t)\varphi(s) ,$$

where $\overline{Y(t)}$ denotes complex conjugate of $Y(t)$.

Following Murata (2001) we describe how the empirical characteristic function can be used in testing for the independence of two components in bivariate distributions.

Given the bivariate sample (X_i, Y_i) , $i = 1, \dots, n$, we are interested in testing for independence of the components X and Y . The test can be based on the following bivariate process,

$$Z_n(t, s) = \sqrt{n} (\varphi_{X,Y}^*(t+s) - \varphi_X^*(t)\varphi_Y^*(s)) ,$$

where $\varphi_{X,Y}^*(t+s) = 1/n \sum_{j=1}^n \exp(itX_j + isY_j)$.

Murata (2001) shows that $Z_n(t, s)$ has Gaussian weak limit and that

$$\text{Var } Z_n(t, s) \approx \left[\varphi_X^*(2t) - (\varphi_X^*(t))^2 \right] \left[\varphi_Y^*(2s) - (\varphi_Y^*(s))^2 \right] , \quad \text{and}$$

$$\text{Cov} \left(Z_n(t, s), \overline{Z_n(t, s)} \right) \approx (1 - |\varphi_X^*(t)|^2) (1 - |\varphi_Y^*(s)|^2) ,$$

The statistics

$$T(t, s) = (\Re Z_n(t, s) \quad \text{Im } Z_n(t, s)) \Sigma^{-1} (\Re Z_n(t, s) \quad \text{Im } Z_n(t, s))'$$

has approximately χ^2 distribution with 2 degrees of freedom for any t and s finite. Symbols \Re and Im stand for the real and imaginary parts of a complex number. The matrix Σ is 2×2 matrix with entries

$$\begin{aligned} \zeta_{11} &= \frac{1}{2} \left[\Re \text{Var} (Z_n(t, s)) + \text{Cov} (Z_n(t, s), \overline{Z_n(t, s)}) \right] \\ \zeta_{12} &= \zeta_{21} = \frac{1}{2} \text{Im} \text{Var} (Z_n(t, s)) , \quad \text{and} \\ \zeta_{22} &= \frac{1}{2} \left[-\Re \text{Var} (Z_n(t, s)) + \text{Cov} (Z_n(t, s), \overline{Z_n(t, s)}) \right] . \end{aligned}$$

Any fixed pair t, s gives a valid test, and in the numerical example we selected $t = 1$ and $s = 1$ for calculational convenience.

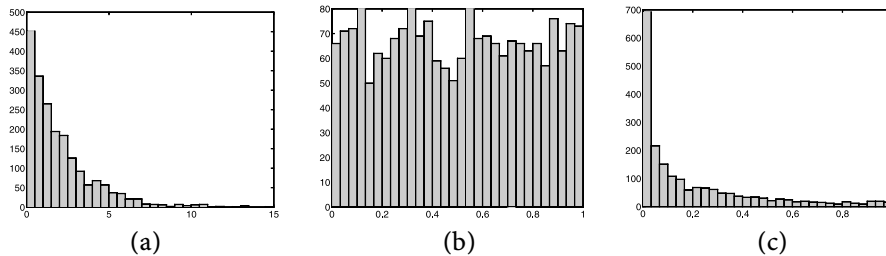


Figure 7.3. (a) Histogram of observed T statistics with theoretical χ^2_2 distribution; (b) p -values of the test when components are independent; and (c) p -values if the test when the second component is a mixture of an independent sample and 3% of the first component

We generated two independent components from the Beta(1, 2) distribution of size $n = 2000$ and found T statistics and corresponding p -values $M = 2000$ times. Figure 7.3a,b depicts histograms of T statistics and p values based on 2000 simulations. Since the generated components X and Y are independent, the histogram for T agrees with asymptotic χ^2_2 distribution, and of course, the p -values are uniform on $[0, 1]$. In Fig. 7.3c we show the p -values when the components X and Y are not independent. Using two independent Beta(1, 2) components X and Y' , the second component Y is constructed as $Y = 0.03X + 0.97Y'$. Notice that for majority of simulational runs the independence hypothesis is rejected, i.e., the p -values cluster around 0.

Fourier and Related Transforms

Functional series have a long history that can be traced back to the early nineteenth century. French mathematician (and politician) Jean-Baptiste-Joseph Fourier, de-

composed a continuous, periodic on $[-\pi, \pi]$ function $f(x)$ into the series of sines and cosines,

$$\frac{a_0}{2} + \sum_{n=1}^{\infty} a_n \cos nx + b_n \sin nx ,$$

where the coefficients a_n and b_n are defined as

$$a_n = \frac{1}{\pi} \int_{-\pi}^{\pi} f(x) \cos nx \, dx , \quad n = 0, 1, 2, \dots$$

$$b_n = \frac{1}{\pi} \int_{-\pi}^{\pi} f(x) \sin nx \, dx , \quad n = 1, 2, \dots .$$

The sequences $\{a_n, n = 0, 1, \dots\}$ and $\{b_n, n = 1, 2, \dots\}$ can be viewed as a transform of the original function f . It is interesting that at the time of Fourier's discovery the very notion of function was not precisely defined. Fourier methods have long history in statistics especially in the theory of nonparametric function and density estimation and characteristic functions.

There are three types of Fourier transforms: integral, serial, and discrete. Next, we focus on discrete transforms and some modifications of the integral transform.

7.2.1 Discrete Fourier Transform

The discrete Fourier transform (DFT) of a sequence $f = \{f_n, n = 0, 1, \dots, N-1\}$ is defined as

$$F = \left\{ \sum_{n=0}^{N-1} f_n w_N^{nk} , \quad k = 0, \dots, N-1 \right\} ,$$

where $w_N = e^{-i2\pi/N}$. The inverse is

$$f = \left\{ \frac{1}{N} \sum_{k=0}^{N-1} F_k w_N^{-nk} , \quad n = 0, \dots, N-1 \right\} .$$

The DFT can be interpreted as the multiplication of the input vector by a matrix; therefore, the discrete Fourier transform is a linear operator. If $Q = \{Q_{nk} = e^{-i2\pi nk}\}_{N \times N}$, then $F = Q \cdot f$. The matrix Q is unitary (up to a scale factor), i.e., $Q^* Q = NI$, where I is the identity matrix and Q^* is the conjugate transpose of Q .

There are many uses of discrete Fourier transform in statistics. It turns cyclic convolutions into component-wise multiplication, and the fast version of DFT has a low computational complexity of $O(n \log(n))$, meaning that the number of operations needed to transform an input of size n is proportional to $n \log(n)$. For a theory and various other uses of DFT in various fields reader is directed to Brigham (1988).

We focus on estimation of a spectral density from the observed data, as an important statistical task in a variety of applied fields in which the information about frequency behavior of the phenomena is of interest.

Let $\{X_t, t \in Z\}$ be a real, weakly stationary time series with zero mean and autocovariance function $\gamma(h) = EX(t+h)X(t)$. An absolutely summable complex-valued function $\gamma(\cdot)$ defined on integers is the autocovariance function of X_t if and only if the function

$$f(\omega) = \frac{1}{2\pi} \sum_{h=-\infty}^{\infty} \gamma(h) e^{-ih\omega} \tag{7.4}$$

is non-negative for all $\omega \in [-\pi, \pi]$. The function $f(\omega)$ is called the spectral density associated with covariance function $\gamma(h)$, and is in fact a version of discrete Fourier transform of the autocovariance function $\gamma(h)$. The spectral density of a stationary process is a symmetric and non-negative function. Given the spectral density, the autocovariance function can uniquely be recovered via inverse Fourier transform,

$$\gamma(h) = \int_{-\pi}^{\pi} f(\omega) e^{ih\omega} d\omega, \quad h = 0, \pm 1, \pm 2, \dots$$

A traditional statistic used as an estimator of the spectral density is the *periodogram*. The periodogram $I(\omega)$, based on a sample X_0, \dots, X_{T-1} is defined as

$$I(\omega_j) = \frac{1}{2\pi T} \left| \sum_{t=0}^{T-1} X_t e^{-it\omega_j} \right|^2, \tag{7.5}$$

where ω_j is the Fourier frequency $\omega_j = 2\pi j/T, j = [-T/2] + 1, \dots, -1, 0, 1, \dots, [T/2]$. By a discrete version of the sampling theorem it holds that $I(\omega)$ is uniquely determined for all $\omega \in [-\pi, \pi]$, given its values at Fourier frequencies.

Calculationally, the periodogram is found by using fast Fourier transform. A simple matlab m-function calculating the periodogram is

```
function out = periodogram(ts)
out = abs(fftshift(fft(ts - mean(ts)))).^2/(2*pi*length(ts));
```

An application of spectral and log-spectral estimation involves famous Wolf's sunspot data set. Although in this situation the statistician does not know the "true" signal, the theory developed by solar scientists helps to evaluate performance of the algorithm.

The Sun's activity peaks every 11 years, creating storms on the surface of our star that disrupt the Earth's magnetic field. These "solar hurricanes" can cause severe problems for electricity transmission systems. An example of influence of such periodic activity to everyday life is 1989 power blackout in the American northeast.

Efforts to monitor the amount and variation of the Sun's activity by counting spots on it have a long and rich history. Relatively complete visual estimates of daily activity date back to 1818, monthly averages can be extrapolated back to 1749, and estimates of annual values can be similarly determined back to 1700. Although Galileo made observations of sunspot numbers in the early 17th century, the modern era of sunspot counting began in the mid-1800s with the research

of Bern Observatory director Rudolph Wolf, who introduced what he called the *Universal Sunspot Number* as an estimate of the solar activity. The square root of Wolf's yearly sunspot numbers are given in Fig. 7.4a, data from Tong (1996), p. 471. Because of wavelet data processing we selected a sample of size a power of two, i.e., only 256 observations from 1733 till 1998. The square root transformation was applied to symmetrize and de-trend the Wolf's counts. Figure 7.4b gives a raw periodogram, while Fig. 7.4c shows the estimator of log-spectral density (Pensky and Vidakovic, 2003).

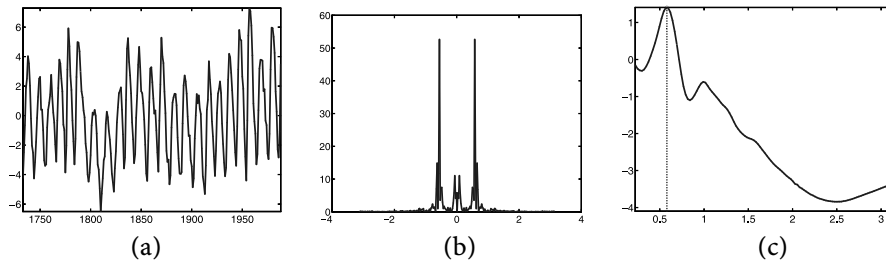


Figure 7.4. (a) Square roots of Wolf's yearly sunspot numbers from 1732–1988 (256 observations); (b) Raw periodogram; (c) An estimator of the log-spectra. The frequency $\omega^* \approx 0.58$ corresponds to Schwabe's period of 10.8 (years)

The estimator reveals a peak at frequency $\omega^* \approx 0.58$, corresponding to the Schwabe's cycle ranging from 9 to 11.5 (years), with an average of $2\pi/0.58 \approx 10.8$ years. The Schwabe cycle is the period between two subsequent maxima or minima the solar activity, although the solar physicists often think in terms of a 22-year magnetic cycle since the sun's magnetic poles reverse direction every 11 years.

7.2.2 Windowed Fourier Transform

Windowed Fourier Transforms are important in providing simultaneous insight in time and frequency behavior of the functions. Standard Fourier Transforms describing the data in the "Fourier domain" are precise in frequency, but not in time. Small changes in the signal (data) at one location cause change in the Fourier domain globally. It was of interest to have transformed domains that are simultaneously precise in both time and frequency domains. Unfortunately, the precision of such an insight is limited by the Heisenberg's Uncertainty Principle.

Suppose $f(t)$ is a signal of finite energy. In mathematical terms, the integral of its modulus squared is finite, or shortly, f belongs to $\mathbb{L}_2(\mathbb{R})$ space.

The integral Fourier transform of the signal

$$\mathcal{F}(f)(\xi) = \hat{f}(\xi) = \int_{\mathbb{R}} f(t) e^{-it\xi} dt, \quad (7.6)$$

describes the allocation of energy content of a signal at different frequencies, but the time-related information is lost.

Windowed Fourier transform (also called short time Fourier transform, STFT) was introduced by Gabor (1946), to measure time-localized frequencies of sound. An atom in Gabor's decomposition is defined via:

$$g_{u,\xi}(t) = e^{i\xi t} g(t - u) ,$$

where g is a real, symmetric, and properly normalized "window" function. [$\|g\| = 1$ so that $\|g_{u,\xi}\| = 1$]

If $f \in \mathbb{L}_2(\mathbb{R})$, then windowed Fourier transform is defined as

$$Sf(u, \xi) = \langle f, g_{u,\xi} \rangle = \int_{\mathbb{R}} f(t)g(t - u) e^{-i\xi t} dt . \tag{7.7}$$

The chief use of windowed Fourier transforms is to analyze time/frequency distribution of signal energy, via a *spectrogram*.

The spectrogram,

$$P_Sf(u, \xi) = |Sf(u, \xi)|^2 = \left| \int_{-\infty}^{\infty} f(t)g(t - u) e^{-i\xi t} dt \right|^2 ,$$

expresses the energy distribution in the signal f , with respect to time and frequency simultaneously.

The following are some basic properties of STFT. Let $f \in \mathbb{L}_2(\mathbb{R}^2)$. Then

$$\text{[Inverse STFT]} \quad f(t) = \frac{1}{2\pi} \int_{\mathbb{R}} \int_{\mathbb{R}} Sf(u, \xi)g(t - u) e^{i\xi t} d\xi du , \tag{7.8}$$

and

$$\text{[Energy Conservation]} \quad \int_{\mathbb{R}} |f(t)|^2 dt = \frac{1}{2\pi} \int_{\mathbb{R}} \int_{\mathbb{R}} |Sf(u, \xi)|^2 d\xi du . \tag{7.9}$$

The following is a characterizing property of STFT:

Let $\Phi \in \mathbb{L}_2(\mathbb{R}^2)$. There exist $f \in \mathbb{L}_2(\mathbb{R}^2)$ such that $\Phi(u, \xi) = Sf(u, \xi)$ if and only if

$$\Phi(u_0, \xi_0) = \frac{1}{2\pi} \int_{\mathbb{R}} \int_{\mathbb{R}} \Phi(u, \xi) \mathbb{K}(u_0, u, \xi_0, \xi) dud\xi , \tag{7.10}$$

where

$$\mathbb{K}(u_0, u, \xi_0, \xi) = \langle g_{u,\xi}, g_{u_0,\xi_0} \rangle = \int_{\mathbb{R}} g(t - u)g(t - u_0) e^{-i(\xi_0 - \xi)t} dt . \tag{7.11}$$

Hilbert Transform

We next describe the Hilbert transform and its use in defining instantaneous frequency, an important measure in statistical analysis of signals.

The Hilbert transform of the function signal $g(t)$ is defined by

$$H_g(t) = \frac{1}{\pi} (VP) \int_{-\infty}^{\infty} \frac{g(\tau)}{t - \tau} d\tau . \tag{7.12}$$

Because of the possible singularity at $\tau = t$, the integral is to be considered as a Cauchy principal value, (VP). From (7.12) we see that $H_g(t)$ is a convolution, $1/(\pi t) * g(t)$.

The spectrum of $H_g(t)$ is related to that of $g(t)$. From the convolution equation,

$$\mathcal{F}(H(t)) = \mathcal{F}\left(\frac{1}{\pi t}\right) \mathcal{F}(g(t)) .$$

where \mathcal{F} is the Fourier transform. With a real signal $g(t)$ one can associate a complex function with the real part equal to $g(t)$ and the imaginary part equal to $H(g(t))$, $h(t) = g(t) - iH(g(t))$.

In statistical signal analysis this associated complex function $h(t)$ is known as analytic signal (or causal signal, since $\hat{h}(\xi) = 0$, for $\xi < 0$). Analytic signals are important since they possess unique phase $\phi(t)$ which leads to the definition of the instantaneous frequency.

If $h(t)$ is represented as $a(t) \cdot \exp\{i\phi(t)\}$, then the quantity $d\phi/dt$ is instantaneous frequency of the signal $g(t)$, at time t . For more discussion and use of instantaneous frequency, the reader is directed to Flandrin (1992, 1999).

7.2.4 Wigner–Ville Transforms

Wigner–Ville Transform (or Distribution) is the method to represent data (signals) in the time/frequency domain. In statistics, Wigner–Ville transform provide a tool to define localized spectral density for the nonstationary processes.

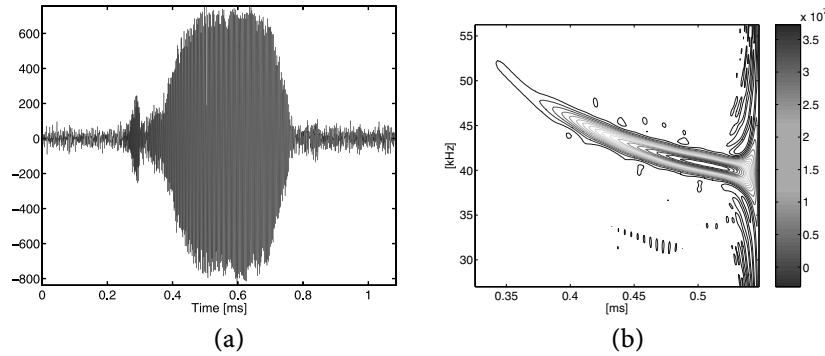


Figure 7.5. (a) Sonar signal from flying bat; (b) its Wigner–Ville transform

Ville (1948) introduced the quadratic form that measures a local time-frequency energy:

$$P_V f(u, \xi) = \int_{\mathbb{R}} f\left(u + \frac{\tau}{2}\right) f^*\left(u - \frac{\tau}{2}\right) e^{-i\tau\xi} d\tau ,$$

where f^* is conjugate of f .

The Wigner–Ville transform is always real since $f(u + \frac{\tau}{2})f^*(u - \frac{\tau}{2})$ has Hermitian symmetry in τ .

Time and frequency are symmetric in $P_V f(u, \xi)$, by applying Parseval formula one gets,

$$P_V f(u, \xi) = \frac{1}{2\pi} \int_{\mathbb{R}} \hat{f}\left(\xi + \frac{Y}{2}\right) \hat{f}^*\left(\xi - \frac{Y}{2}\right) e^{-iyu} dY, \quad (7.13)$$

For any $f \in \mathbb{L}_2(\mathbb{R})$

$$\int_{\mathbb{R}} P_V f(u, \xi) du = |\hat{f}(\xi)|^2, \quad (7.14)$$

i.e., the time marginalization reproduces power spectrum, and

$$\int_{\mathbb{R}} P_V f(u, \xi) d\xi = 2\pi |f(u)|^2, \quad (7.15)$$

i.e, the frequency marginalization is proportional to the squared modulus of the signal.

Integral (7.13) states that one-dimensional Fourier transform of $g_\xi(u) = P_V f(u, \xi)$, with respect to u is,

$$\hat{g}_\xi(\gamma) = \hat{f}\left(\xi + \frac{Y}{2}\right) \hat{f}^*\left(\xi - \frac{Y}{2}\right).$$

If $\gamma = 0$, $\hat{g}_\xi(0) = \int_{\mathbb{R}} g_\xi(u) du$, which proves (7.14). Similarly for (7.15).

For example,

(i) if $f(t) = \mathbf{1}(-T \leq t \leq T)$, then

$$P_V f(u, \xi) = \frac{2 \sin[2(T - |u|)\xi]}{\xi} \mathbf{1}(-T \leq u \leq T).$$

Plot $P_V f(u, \xi)$.

(ii) if $f(t) = \exp\{i\lambda(t + \alpha t^2/2)\}$, then $P_V(u, \xi) = 2\pi\delta(\xi - \lambda(1 + \alpha u))$.

(iii) a Gaussian $f(t) = (\sigma^2\pi)^{-1/4} \exp(-t^2/(2\sigma^2))$ is transformed into

$$P_V f(u, \xi) = \frac{1}{\pi} \exp\left(-\frac{u^2}{\sigma^2} - \sigma^2 \xi^2\right).$$

In this case, $P_V f(u, \xi) = |f(u)|^2 \cdot |\hat{f}(\xi)|^2$. The Gaussian is the only (up to time and frequency shifts) distribution for which Wigner–Ville transform remains positive. Some basic properties of Wigner–Ville transforms are listed in Table 7.1.

Next we show that expected value of Wigner–Ville transform of a random process can serve as a definition for generalized spectrum of a non-stationary process. Let $X(t)$ be real-valued zero-mean random process with covariance function

$$EX(t)X(s) = R(t, s) = R\left(u + \frac{\tau}{2}, u - \frac{\tau}{2}\right) = C(u, \tau),$$

after substitution $\tau = t - s$ and $u = (t + s)/2$.

Table 7.1. Properties of Wigner–Ville transform

Function	Wigner–Ville
$f(t)$	$P_V f(u, \xi)$
$e^{i\phi} f(t)$	$P_V f(u, \xi)$
$f(t - u_0)$	$P_V f(u - u_0, \xi)$
$e^{i\xi_0 t} f(t)$	$P_V f(u, \xi - \xi_0)$
$e^{iat^2} f(t)$	$P_V f(u, \xi - 2au)$
$\frac{1}{\sqrt{s}} f(t/s)$	$P_V f(u/s, s\xi)$

Now, if the process $X(t)$ is stationary, then $C(u, \tau)$ is a function of τ only and

$$P_X(\xi) = \int_{-\infty}^{\infty} C(\tau) e^{-i\xi\tau} d\tau$$

is its power spectrum.

For arbitrary process Flandrin (1999) defined “power spectrum” as

$$P_X(\xi) = \int_{-\infty}^{\infty} C(u, \tau) e^{-i\xi\tau} d\tau .$$

Thus, $P_X(\xi)$ can be represented as $E P_V X(u, \xi)$, where

$$P_V X(u, \xi) = \int_{-\infty}^{\infty} X\left(u + \frac{\tau}{2}\right) X\left(u - \frac{\tau}{2}\right) e^{-i\xi\tau} d\tau .$$

For more information on Wigner–Ville transforms and their statistical use the reader is directed to Baraniuk (1994), Carmona, Hwang and Torresani (1998) Flandrin (1999), and Mallat (1999), among others.

Wavelets and Other Multiscale Transforms

7.3

Given their recent popularity and clear evidence of wide applicability the most of the space in this chapter is devoted to Wavelet transforms. Statistical multiscale modeling has, in recent decade, become a well established area in both theoretical and applied statistics, with impact to developments in statistical methodology.

Wavelet-based methods are important in statistics in areas such as regression, density and function estimation, factor analysis, modeling and forecasting in time series analysis, in assessing self-similarity and fractality in data, in spatial statistics.

The attention of the statistical community was attracted in late 1980’s and early 1990’s, when Donoho, Johnstone, and their coauthors demonstrated that wavelet thresholding, a simple denoising procedure, had desirable statistical optimality properties. Since then, wavelets have proved useful in many statistical disciplines,

notably in nonparametric statistics and time series analysis. Bayesian concepts and modeling approaches have, more recently, been identified as providing promising contexts for wavelet-based denoising applications.

In addition to replacing traditional orthonormal bases in a variety of statistical problems, wavelets brought novel techniques and invigorated some of the existing ones.

A Case Study

7.3.1

We start first with a statistical application of wavelet transforms. This example emphasizes the specificity of wavelet-based denoising not shared by standard state-of-the-art denoising techniques.

A researcher in geology was interested in predicting earthquakes by the level of water in nearby wells. She had a large ($8192 = 2^{13}$ measurements) data set of water levels taken every hour in a period of time of about one year in a California well. Here is the description of the problem.

The ability of water wells to act as strain meters has been observed for centuries. The Chinese, for example, have records of water flowing from wells prior to earthquakes. Lab studies indicate that a seismic slip occurs along a fault prior to rupture. Recent work has attempted to quantify this response, in an effort to use water wells as sensitive indicators of volumetric strain. If this is possible, water wells could aid in earthquake prediction by sensing precursory earthquake strain.

We have water level records from six wells in southern California, collected over a six-year time span. At least 13 moderate-size earthquakes (magnitude 4.0–6.0) occurred in close proximity to the wells during this time interval. There is a significant amount of noise in the water level record which must first be filtered out. Environmental factors such as earth tides and atmospheric pressure create noise with frequencies ranging from seasonal to semidiurnal. The amount of rainfall also affects the water level, as do surface loading, pumping, recharge (such as an increase in water level due to irrigation), and sonic booms, to name a few. Once the noise is subtracted from the signal, the record can be analyzed for changes in water level, either an increase or a decrease depending upon whether the aquifer is experiencing a tensile or compressional volume strain, just prior to an earthquake.

A plot of the raw data for hourly measurements over one year ($8192 = 2^{13}$ observations) is given in Fig. 7.6a, with a close-up in Fig. 7.6b. After applying the wavelet transform and further processing the wavelet coefficients (thresholding), we obtained a fairly clean signal with a big jump at the earthquake time. The wavelet-denoised data are given in Fig. 7.6d. The magnitude of the water level change at the earthquake time did not get distorted in contrast to traditional smoothing techniques. This local adaptivity is a desirable feature of wavelet methods.

For example, Fig. 7.6c, is denoised signal after applying `supsmo` smoothing procedure. Note that the earthquake jump is smoothed, as well.

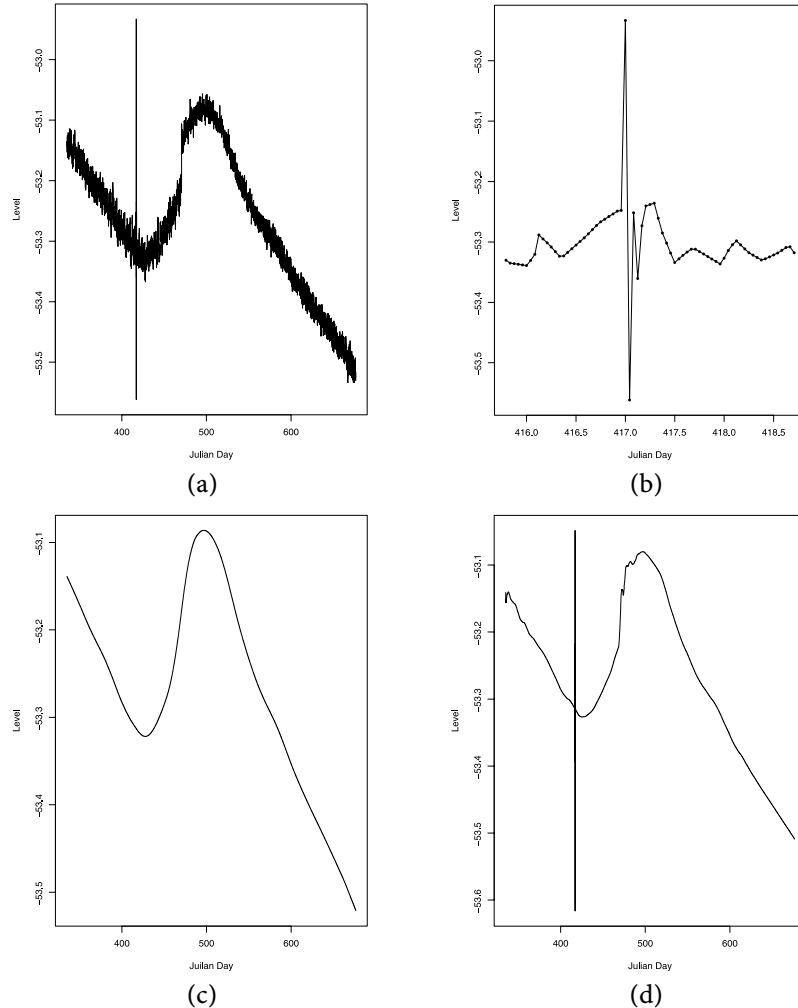


Figure 7.6. (a) shows $n = 8192$ hourly measurements of the water level for a well in an earthquake zone. Notice the wide range of water levels at the time of an earthquake around $t = 417$. (b) focuses on the data around the earthquake time. (c) demonstrates action of a standard smoother `supsmo`, and (d) gives a wavelet based reconstruction

7.3.2 Continuous Wavelet Transform

The first theoretical results in wavelets had been concerned with continuous wavelet decompositions of functions and go back to the early 1980s. Papers of Morlet et al. (1982) and Grossmann and Morlet (1984, 1985) were among the first on this subject.

Let $\psi_{a,b}(x)$, $a \in \mathbb{R} \setminus \{0\}$, $b \in \mathbb{R}$ be a family of functions defined as translations and re-scales of a single function $\psi(x) \in \mathbb{L}_2(\mathbb{R})$,

$$\psi_{a,b}(x) = \frac{1}{\sqrt{|a|}} \psi\left(\frac{x-b}{a}\right). \tag{7.16}$$

Normalization constant $1/\sqrt{|a|}$ ensures that the norm $\|\psi_{a,b}(x)\|$ is independent of a and b . The function ψ (called *the wavelet function*) is assumed to satisfy the *admissibility condition*,

$$C_\psi = \int_{\mathbb{R}} \frac{|\Psi(\omega)|^2}{|\omega|} d\omega < \infty, \tag{7.17}$$

where $\Psi(\omega) = \int_{\mathbb{R}} \psi(x) e^{-ix\omega} dx$ is the Fourier transform of $\psi(x)$. The admissibility condition (7.17) implies

$$0 = \Psi(0) = \int \psi(x) dx.$$

Also, if $\int \psi(x) dx = 0$ and $\int (1 + |x|^\alpha) |\psi(x)| dx < \infty$ for some $\alpha > 0$, then $C_\psi < \infty$.

Wavelet functions are usually normalized to “have unit energy”, i.e., $\|\psi_{a,b}(x)\| = 1$.

For example, the second derivative of the Gaussian function,

$$\psi(x) = \frac{d^2}{dx^2} [-C e^{-x^2/2}] = C (1 - x^2) e^{-x^2/2}, \quad C = \frac{2}{\sqrt{3\sqrt{\pi}}},$$

is an example of an admissible wavelet, called Mexican Hat or Marr’s wavelet, see Fig. 7.7.

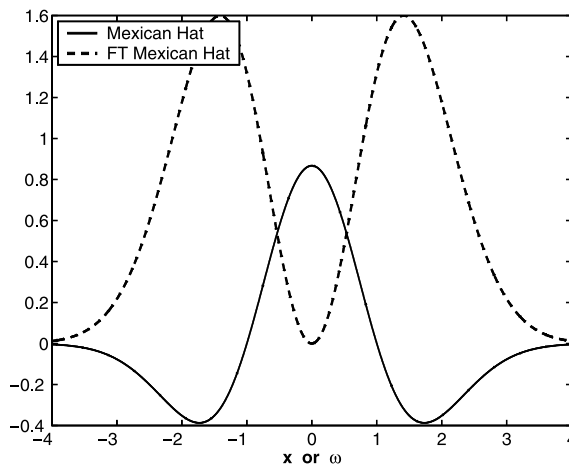


Figure 7.7. Mexican hat wavelet (solid line) and its Fourier transform (dashed line)

For any square integrable function $f(x)$, the continuous wavelet transform is defined as a function of two variables

$$\mathcal{CWT}_f(a, b) = \langle f, \psi_{a,b} \rangle = \int f(x) \overline{\psi_{a,b}(x)} dx.$$

Here the dilation and translation parameters, a and b , respectively, vary continuously over $\mathbb{R} \setminus \{0\} \times \mathbb{R}$.

Figure 7.8 gives the doppler test function, $f = 1/(t + 0.05) \sqrt{t(1-t)} \sin(2\pi \cdot 1.05t)$, $0 \leq t \leq 1$, and its continuous wavelet transform. The wavelet used was Mexican Hat. Notice the distribution of “energy” in the time/frequency plane in Fig. 7.8b.

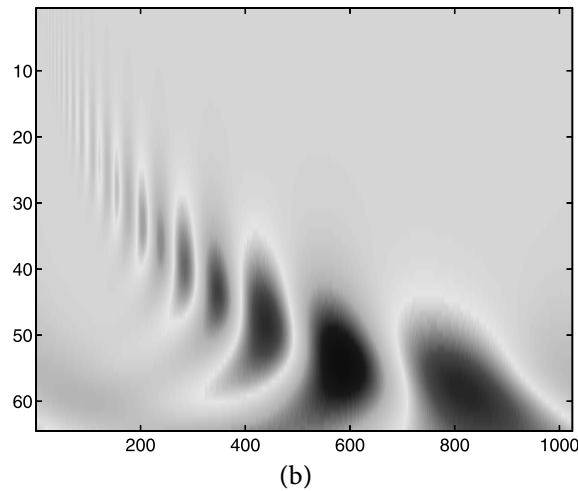
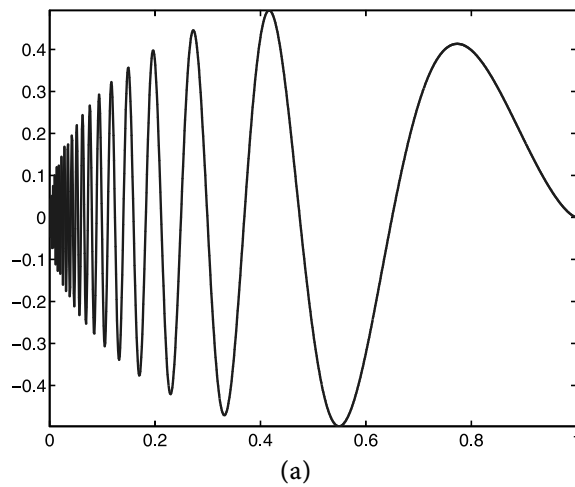


Figure 7.8. (a) Doppler signal; (b) Continuous wavelet transform of doppler signal by the Mexican hat wavelet

Resolution of Identity. When the admissibility condition is satisfied, i.e., $C_\psi < \infty$, it is possible to find the inverse continuous transform via the relation known as *resolution of identity* or *Calderón's reproducing identity*,

$$f(x) = \frac{1}{C_\psi} \int_{\mathbb{R}^2} \mathcal{C} \mathcal{W} \mathcal{T}_f(a, b) \psi_{a,b}(x) \frac{da db}{a^2} .$$

The continuous wavelet transform of a function of one variable is a function of two variables. Clearly, the transform is redundant. To “minimize” the transform one can select discrete values of a and b and still have a lossless transform. This is achieved by so called *critical sampling*.

The critical sampling defined by

$$a = 2^{-j} , \quad b = k2^{-j} , \quad j, k \in \mathbb{Z} , \tag{7.18}$$

will produce the minimal, but complete basis. Any coarser sampling will not produce a unique inverse transform. Moreover under mild conditions on the wavelet function ψ , such sampling produces an orthogonal basis $\{\psi_{jk}(x) = 2^{j/2} \psi(2^j x - k), j, k \in \mathbb{Z}\}$. To formally describe properties of minimal and orthogonal wavelet bases a multiresolution formalism is needed.

Multiresolution Analysis

7.3.3

Fundamental for construction of critically sampled orthogonal wavelets is a notion of multiresolution analysis introduced by Mallat (1989a, 1989b). A multiresolution analysis (MRA) is a sequence of closed subspaces $V_n, n \in \mathbb{Z}$ in $\mathbb{L}_2(\mathbb{R})$ such that they lie in a containment hierarchy

$$\cdots \subset V_{-2} \subset V_{-1} \subset V_0 \subset V_1 \subset V_2 \subset \cdots . \tag{7.19}$$

The nested spaces have an intersection that contains only the zero function and a union that contains all square integrable functions.

$$\bigcap_n V_j = \{\mathbf{0}\} , \quad \overline{\bigcup_j V_j} = \mathbb{L}_2(\mathbb{R}) .$$

(With \overline{A} we denoted the closure of a set A). The hierarchy (7.19) is constructed such that V -spaces are self-similar,

$$f(2^j x) \in V_j \quad \text{iff} \quad f(x) \in V_0 . \tag{7.20}$$

with the requirement that there exists a *scaling function* $\phi \in V_0$ whose integer-translates span the space V_0 ,

$$V_0 = \left\{ f \in \mathbb{L}_2(\mathbb{R}) \mid f(x) = \sum_k c_k \phi(x - k) \right\} ,$$

and for which the family $\{\phi(\bullet - k), k \in \mathbb{Z}\}$ is an orthonormal basis. It can be assumed that $\int \phi(x) dx \geq 0$. With this assumption this integral is in fact equal to 1.

Because of containment $V_0 \subset V_1$, the function $\phi(x) \in V_0$ can be represented as a linear combination of functions from V_1 , i.e.,

$$\phi(x) = \sum_{k \in \mathbb{Z}} h_k \sqrt{2} \phi(2x - k), \quad (7.21)$$

for some coefficients $h_k, k \in \mathbb{Z}$. This equation called the *scaling equation* (or two-scale equation) is fundamental in constructing, exploring, and utilizing wavelets.

2 Theorem 2 For the scaling function it holds

$$\int_{\mathbb{R}} \phi(x) dx = 1,$$

or, equivalently,

$$\Phi(0) = 1,$$

where $\Phi(\omega)$ is Fourier transform of ϕ , $\int_{\mathbb{R}} \phi(x) e^{-i\omega x} dx$.

The coefficients h_n in (7.21) are important in efficient application of wavelet transforms. The (possibly infinite) vector $\mathbf{h} = \{h_n, n \in \mathbb{Z}\}$ will be called a *wavelet filter*. It is a low-pass (averaging) filter as will become clear later by its analysis in the Fourier domain.

To further explore properties of multiresolution analysis subspaces and their bases, we will often work in the Fourier domain.

It will be convenient to use Fourier domain for subsequent analysis of wavelet paradigm. Define the function m_0 as follows:

$$m_0(\omega) = \frac{1}{\sqrt{2}} \sum_{k \in \mathbb{Z}} h_k e^{-ik\omega} = \frac{1}{\sqrt{2}} H(\omega). \quad (7.22)$$

The function in (7.22) is sometimes called the *transfer function* and it describes the behavior of the associated filter \mathbf{h} in the Fourier domain. Notice that the function m_0 is 2π -periodic and that filter taps $\{h_n, n \in \mathbb{Z}\}$ are in fact the Fourier coefficients in the Fourier series of $H(\omega) = \sqrt{2}m_0(\omega)$.

In the Fourier domain the relation (7.21) becomes

$$\Phi(\omega) = m_0\left(\frac{\omega}{2}\right) \Phi\left(\frac{\omega}{2}\right), \quad (7.23)$$

where $\Phi(\omega)$ is the Fourier transform of $\phi(x)$. Indeed,

$$\begin{aligned} \Phi(\omega) &= \int_{-\infty}^{\infty} \phi(x) e^{-i\omega x} dx \\ &= \sum_k \sqrt{2} h_k \int_{-\infty}^{\infty} \phi(2x - k) e^{-i\omega x} dx \end{aligned}$$

$$\begin{aligned}
 &= \sum_k \frac{h_k}{\sqrt{2}} e^{-ik\omega/2} \int_{-\infty}^{\infty} \phi(2x-k) e^{-i(2x-k)\omega/2} d(2x-k) \\
 &= \sum_k \frac{h_k}{\sqrt{2}} e^{-ik\omega/2} \Phi\left(\frac{\omega}{2}\right) \\
 &= m_0\left(\frac{\omega}{2}\right) \Phi\left(\frac{\omega}{2}\right).
 \end{aligned}$$

By iterating (7.23), one gets

$$\Phi(\omega) = \prod_{n=1}^{\infty} m_0\left(\frac{\omega}{2^n}\right), \quad (7.24)$$

which is convergent under very mild conditions concerning the rates of decay of the scaling function ϕ .

Next, we prove two important properties of wavelet filters associated with an orthogonal multiresolution analysis, *normalization* and *orthogonality*.

Normalization.

$$\sum_{k \in \mathbb{Z}} h_k = \sqrt{2}. \quad (7.25)$$

Proof:

$$\begin{aligned}
 \int \phi(x) dx &= \sqrt{2} \sum_k h_k \int \phi(2x-k) dx \\
 &= \sqrt{2} \sum_k h_k \frac{1}{2} \int \phi(2x-k) d(2x-k) \\
 &= \frac{\sqrt{2}}{2} \sum_k h_k \int \phi(x) dx.
 \end{aligned}$$

Since $\int \phi(x) dx \neq 0$ by assumption, (7.25) follows.

This result also follows from $m_0(0) = 1$.

Orthogonality. For any $l \in \mathbb{Z}$,

$$\sum_k h_k h_{k-2l} = \delta_l. \quad (7.26)$$

Proof: Notice first that from the scaling equation (7.21) it follows that

$$\begin{aligned}
 \phi(x)\phi(x-l) &= \sqrt{2} \sum_k h_k \phi(2x-k)\phi(x-l) \\
 &= \sqrt{2} \sum_k h_k \phi(2x-k) \sqrt{2} \sum_m h_m \phi(2(x-l)-m).
 \end{aligned} \quad (7.27)$$

By integrating the both sides in (7.27) we obtain

$$\begin{aligned}\delta_l &= 2 \sum_k h_k \left[\sum_m h_m \frac{1}{2} \int \phi(2x-k)\phi(2x-2l-m) d(2x) \right] \\ &= \sum_k \sum_m h_k h_m \delta_{k,2l+m} \\ &= \sum_k h_k h_{k-2l}.\end{aligned}$$

The last line is obtained by taking $k = 2l + m$.

An important special case is $l = 0$ for which (7.26) becomes

$$\sum_k h_k^2 = 1. \quad (7.28)$$

The fact that the system $\{\phi(\bullet - k), k \in \mathbb{Z}\}$ constitutes an orthonormal basis for V_0 can be expressed in the Fourier domain in terms of either $\Phi(\omega)$ or $m_0(\omega)$.

In terms of $\Phi(\omega)$,

$$\sum_{l=-\infty}^{\infty} |\Phi(\omega + 2\pi l)|^2 = 1. \quad (7.29)$$

From the Plancherel identity and the 2π -periodicity of $e^{i\omega k}$ it follows

$$\begin{aligned}\delta_k &= \int_{\mathbb{R}} \phi(x) \overline{\phi(x-k)} dx \\ &= \frac{1}{2\pi} \int_{\mathbb{R}} \Phi(\omega) \overline{\Phi(\omega)} e^{i\omega k} d\omega \\ &= \frac{1}{2\pi} \int_0^{2\pi} \sum_{l=-\infty}^{\infty} |\Phi(\omega + 2\pi l)|^2 e^{i\omega k} d\omega.\end{aligned} \quad (7.30)$$

The last line in (7.30) is the Fourier coefficient a_k in the Fourier series decomposition of

$$f(\omega) = \sum_{l=-\infty}^{\infty} |\Phi(\omega + 2\pi l)|^2.$$

Due to the uniqueness of Fourier representation, $f(\omega) = 1$. As a side result, we obtain that $\Phi(2\pi n) = 0$, $n \neq 0$, and $\sum_n \phi(x-n) = 1$. The last result follows from inspection of coefficients c_k in the Fourier decomposition of $\sum_n \phi(x-n)$, the series $\sum_k c_k e^{2\pi i k x}$. As this function is 1-periodic,

$$c_k = \int_0^1 \left(\sum_n \phi(x-n) \right) e^{-2\pi i k x} dx = \int_{-\infty}^{\infty} \phi(x) e^{-2\pi i k x} dx = \Phi(2\pi k) = \delta_{0,k}.$$

Remark 1 Utilizing the identity (7.29), any set of independent functions spanning $V_0, \{\phi(x-k), k \in \mathbb{Z}\}$, can be orthogonalized in the Fourier domain. The orthonormal basis is generated by integer-shifts of the function

$$\mathcal{F}^{-1} \left[\frac{\Phi(\omega)}{\sqrt{\sum_{l=-\infty}^{\infty} |\Phi(\omega + 2\pi l)|^2}} \right]. \tag{7.31}$$

This normalization in the Fourier domain is used in constructing of some wavelet bases.

Orthogonality condition 7.29 can be expressed in terms of m_0 as:

$$|m_0(\omega)|^2 + |m_0(\omega + \pi)|^2 = 1. \tag{7.32}$$

Since $\sum_{l=-\infty}^{\infty} |\Phi(\omega + 2l\pi)|^2 = 1$, then by (7.23)

$$\sum_{l=-\infty}^{\infty} |m_0(\omega + l\pi)|^2 |\Phi(\omega + l\pi)|^2 = 1. \tag{7.33}$$

Now split the sum in (7.33) into two sums – one with odd and the other with even indices, i.e.,

$$1 = \sum_{k=-\infty}^{\infty} |m_0(\omega + 2k\pi)|^2 |\Phi(\omega + 2k\pi)|^2 + \sum_{k=-\infty}^{\infty} |m_0(\omega + (2k + 1)\pi)|^2 |\Phi(\omega + (2k + 1)\pi)|^2.$$

To simplify the above expression, we use (7.29) and the 2π -periodicity of $m_0(\omega)$.

$$\begin{aligned} 1 &= |m_0(\omega)|^2 \sum_{k=-\infty}^{\infty} |\Phi(\omega + 2k\pi)|^2 + |m_0(\omega + \pi)|^2 \sum_{k=-\infty}^{\infty} |\Phi((\omega + \pi) + 2k\pi)|^2 \\ &= |m_0(\omega)|^2 + |m_0(\omega + \pi)|^2. \end{aligned}$$

Whenever a sequence of subspaces satisfies MRA properties, there exists (though not unique) an orthonormal basis for $\mathbb{L}_2(\mathbb{R})$,

$$\left\{ \psi_{jk}(x) = 2^{j/2} \psi(2^j x - k), \quad j, k \in \mathbb{Z} \right\} \tag{7.34}$$

such that $\{\psi_{jk}(x), j\text{-fixed}, k \in \mathbb{Z}\}$ is an orthonormal basis of the “difference space” $W_j = V_{j+1} \ominus V_j$. The function $\psi(x) = \psi_{00}(x)$ is called a *wavelet function* or informally *the mother wavelet*.

Next, we discuss the derivation of a wavelet function from the scaling function. Since $\psi(x) \in V_1$ (because of the containment $W_0 \subset V_1$), it can be represented as

$$\psi(x) = \sum_{k \in \mathbb{Z}} g_k \sqrt{2} \phi(2x - k), \quad (7.35)$$

for some coefficients g_k , $k \in \mathbb{Z}$.

Define

$$m_1(\omega) = \frac{1}{\sqrt{2}} \sum_k g_k e^{-ik\omega}. \quad (7.36)$$

By mimicking what was done with m_0 , we obtain the Fourier counterpart of (7.35),

$$\Psi(\omega) = m_1\left(\frac{\omega}{2}\right) \Phi\left(\frac{\omega}{2}\right). \quad (7.37)$$

The spaces W_0 and V_0 are orthogonal by construction. Therefore,

$$\begin{aligned} 0 &= \int \psi(x) \phi(x - k) dx = \frac{1}{2\pi} \int \Psi(\omega) \overline{\Phi(\omega)} e^{i\omega k} d\omega \\ &= \frac{1}{2\pi} \int_0^{2\pi} \sum_{l=-\infty}^{\infty} \Psi(\omega + 2l\pi) \overline{\Phi(\omega + 2l\pi)} e^{i\omega k} d\omega. \end{aligned}$$

By repeating the Fourier series argument, as in (7.29), we conclude

$$\sum_{l=-\infty}^{\infty} \Psi(\omega + 2l\pi) \overline{\Phi(\omega + 2l\pi)} = 0.$$

By taking into account the definitions of m_0 and m_1 , and by the derivation as in (7.32), we find

$$m_1(\omega) \overline{m_0(\omega)} + m_1(\omega + \pi) \overline{m_0(\omega + \pi)} = 0. \quad (7.38)$$

From (7.38), we conclude that there exists a function $\lambda(\omega)$ such that

$$(m_1(\omega), m_1(\omega + \pi)) = \lambda(\omega) \left(\overline{m_0(\omega + \pi)}, -\overline{m_0(\omega)} \right). \quad (7.39)$$

By substituting $\xi = \omega + \pi$ and by using the 2π -periodicity of m_0 and m_1 , we conclude that

$$\lambda(\omega) = -\lambda(\omega + \pi), \quad \text{and} \quad (7.40)$$

$\lambda(\omega)$ is 2π -periodic.

Any function $\lambda(\omega)$ of the form $e^{\pm i\omega} S(2\omega)$, where S is an $\mathbb{L}_2([0, 2\pi])$, 2π -periodic function, will satisfy (7.38); however, only the functions for which $|\lambda(\omega)| = 1$ will define an orthogonal basis ψ_{jk} of $\mathbb{L}_2(\mathbb{R})$.

To summarize, we choose $\lambda(\omega)$ such that

- (i) $\lambda(\omega)$ is 2π -periodic,
- (ii) $\lambda(\omega) = -\lambda(\omega + \pi)$, and
- (iii) $|\lambda(\omega)|^2 = 1$.

Standard choices for $\lambda(\omega)$ are $-e^{-i\omega}$, $e^{-i\omega}$, and $e^{i\omega}$; however, any other function satisfying (i)–(iii) will generate a valid m_1 . We choose to define $m_1(\omega)$ as

$$m_1(\omega) = -e^{-i\omega} \overline{m_0(\omega + \pi)}. \tag{7.41}$$

since it leads to a convenient and standard connection between the filters \mathbf{h} and \mathbf{g} .

The form of m_1 and (7.29) imply that $\{\psi(\bullet - k), k \in \mathbb{Z}\}$ is an orthonormal basis for W_0 .

Since $|m_1(\omega)| = |m_0(\omega + \pi)|$, the orthogonality condition (7.32) can be rewritten as

$$|m_0(\omega)|^2 + |m_1(\omega)|^2 = 1. \tag{7.42}$$

By comparing the definition of m_1 in (7.36) with

$$\begin{aligned} m_1(\omega) &= -e^{-i\omega} \frac{1}{\sqrt{2}} \sum_k h_k e^{i(\omega+\pi)k} \\ &= \frac{1}{\sqrt{2}} \sum_k (-1)^{1-k} h_k e^{-i\omega(1-k)} \\ &= \frac{1}{\sqrt{2}} \sum_n (-1)^n h_{1-n} e^{-i\omega n}, \end{aligned}$$

we relate g_n and h_n as

$$g_n = (-1)^n h_{1-n}. \tag{7.43}$$

In signal processing literature, (7.43) is known as the *quadrature mirror relation* and the filters \mathbf{h} and \mathbf{g} as *quadrature mirror filters*.

Remark 2 Choosing $\lambda(\omega) = e^{i\omega}$ leads to the rarely used high-pass filter $g_n = (-1)^{n-1} h_{1-n}$. It is convenient to define g_n as $(-1)^n h_{1-n+M}$, where M is a “shift constant.” Such re-indexing of \mathbf{g} affects only the shift-location of the wavelet function.

2

Haar Wavelets

7.3.4

In addition to their simplicity and formidable applicability, Haar wavelets have tremendous educational value. Here we illustrate some of the relations discussed

in the Sect. 7.3.3 using the Haar wavelet. We start with scaling function $\phi(x) = 1(0 \leq x \leq 1)$ and pretend that everything else is unknown. By inspection of simple graphs of two scaled Haar wavelets $\phi(2x)$ and $\phi(2x + 1)$ stuck to each other, we conclude that the scaling equation (7.21) is

$$\begin{aligned}\phi(x) &= \phi(2x) + \phi(2x - 1) \\ &= \frac{1}{\sqrt{2}}\sqrt{2}\phi(2x) + \frac{1}{\sqrt{2}}\sqrt{2}\phi(2x - 1),\end{aligned}\quad (7.44)$$

which yields the wavelet filter coefficients:

$$h_0 = h_1 = \frac{1}{\sqrt{2}}.$$

The transfer functions are

$$m_0(\omega) = \frac{1}{\sqrt{2}} \left(\frac{1}{\sqrt{2}} e^{-i\omega 0} \right) + \frac{1}{\sqrt{2}} \left(\frac{1}{\sqrt{2}} e^{-i\omega 1} \right) = \frac{1 + e^{-i\omega}}{2}.$$

and

$$m_1(\omega) = -e^{-i\omega} \overline{m_0(\omega + \pi)} = -e^{-i\omega} \left(\frac{1}{2} - \frac{1}{2} e^{i\omega} \right) = \frac{1 - e^{-i\omega}}{2}.$$

Notice that $m_0(\omega) = |m_0(\omega)| e^{i\varphi(\omega)} = \cos(\omega/2) \cdot e^{-i\omega/2}$ (after $\cos x = (e^{ix} + e^{-ix})/2$). Since $\varphi(\omega) = -\frac{\omega}{2}$, the Haar wavelet has *linear phase*, i.e., the scaling function is symmetric in the time domain. The orthogonality condition $|m_0(\omega)|^2 + |m_1(\omega)|^2 = 1$ is easily verified, as well.

Relation (7.37) becomes

$$\Psi(\omega) = \frac{1 - e^{-i\omega/2}}{2} \Phi\left(\frac{\omega}{2}\right) = \frac{1}{2} \Phi\left(\frac{\omega}{2}\right) - \frac{1}{2} \Phi\left(\frac{\omega}{2}\right) e^{-i\omega/2},$$

and by applying the inverse Fourier transform we obtain

$$\psi(x) = \phi(2x) - \phi(2x - 1)$$

in the time-domain. Therefore we “have found” the Haar wavelet function ψ . From the expression for m_1 or by inspecting the representation of $\psi(x)$ by $\phi(2x)$ and $\phi(2x - 1)$, we “conclude” that $g_0 = -g_{-1} = \frac{1}{\sqrt{2}}$.

Although the Haar wavelets are well localized in the time domain, in the frequency domain they decay at the slow rate of $O(1/n)$ and are not effective in approximating smooth functions.

7.3.5 Daubechies' Wavelets

The most important family of wavelets was discovered by Ingrid Daubechies and fully described in Daubechies (1992). This family is compactly supported with various degrees of smoothness.

The formal derivation of Daubechies' wavelets goes beyond the scope of this chapter, but the filter coefficients of some of its family members can be found by following considerations.

For example, to derive the filter taps of a wavelet with N vanishing moments, or equivalently, $2N$ filter taps, we use the following equations.

The normalization property of scaling function implies

$$\sum_{i=0}^{2N-1} h_i = \sqrt{2},$$

requirement for vanishing moments for wavelet function ψ leads to

$$\sum_{i=0}^{2N-1} (-1)^i t^k h_i = 0, \quad k = 0, 1, \dots, N-1,$$

and, finally, the orthogonality property can be expressed as

$$\sum_{i=0}^{2N-1} h_i h_{i+2k} = \delta_k \quad k = 0, 1, \dots, N-1.$$

We obtained $2N+1$ equations with $2N$ unknowns; however the system is solvable since the equations are not linearly independent.

Example 4 For $N = 2$, we obtain the system:

4

$$\begin{cases} h_0 + h_1 + h_2 + h_3 = \sqrt{2} \\ h_0^2 + h_1^2 + h_2^2 + h_3^2 = 1 \\ -h_1 + 2h_2 - 3h_3 = 0, \\ h_0 h_2 + h_1 h_3 = 0 \end{cases},$$

which has a solution $h_0 = \frac{1+\sqrt{3}}{4\sqrt{2}}$, $h_1 = \frac{3+\sqrt{3}}{4\sqrt{2}}$, $h_2 = \frac{3-\sqrt{3}}{4\sqrt{2}}$, and $h_3 = \frac{1-\sqrt{3}}{4\sqrt{2}}$.

For $N = 4$, the system is

$$\begin{cases} h_0 + h_1 + h_2 + h_3 + h_4 + h_5 + h_6 + h_7 = \sqrt{2} \\ h_0^2 + h_1^2 + h_2^2 + h_3^2 + h_4^2 + h_5^2 + h_6^2 + h_7^2 = 1 \\ h_0 - h_1 + h_2 - h_3 + h_4 - h_5 + h_6 - h_7 = 0 \\ h_0 h_2 + h_1 h_3 + h_2 h_4 + h_3 h_5 + h_4 h_6 + h_5 h_7 = 0 \\ h_0 h_4 + h_1 h_5 + h_2 h_6 + h_3 h_7 = 0 \\ h_0 h_6 + h_1 h_7 = 0 \\ 0h_0 - 1h_1 + 2h_2 - 3h_3 + 4h_4 - 5h_5 + 6h_6 - 7h_7 = 0 \\ 0h_0 - 1h_1 + 4h_2 - 9h_3 + 16h_4 - 25h_5 + 36h_6 - 49h_7 = 0 \\ 0h_0 - 1h_1 + 8h_2 - 27h_3 + 64h_4 - 125h_5 + 216h_6 - 343h_7 = 0. \end{cases}$$

Figure 7.9 depicts two scaling function and wavelet pairs from the Daubechies family. Figure 7.9a,b depict the pair with two vanishing moments, while Fig. 7.9c,d depict the pair with four vanishing moments.

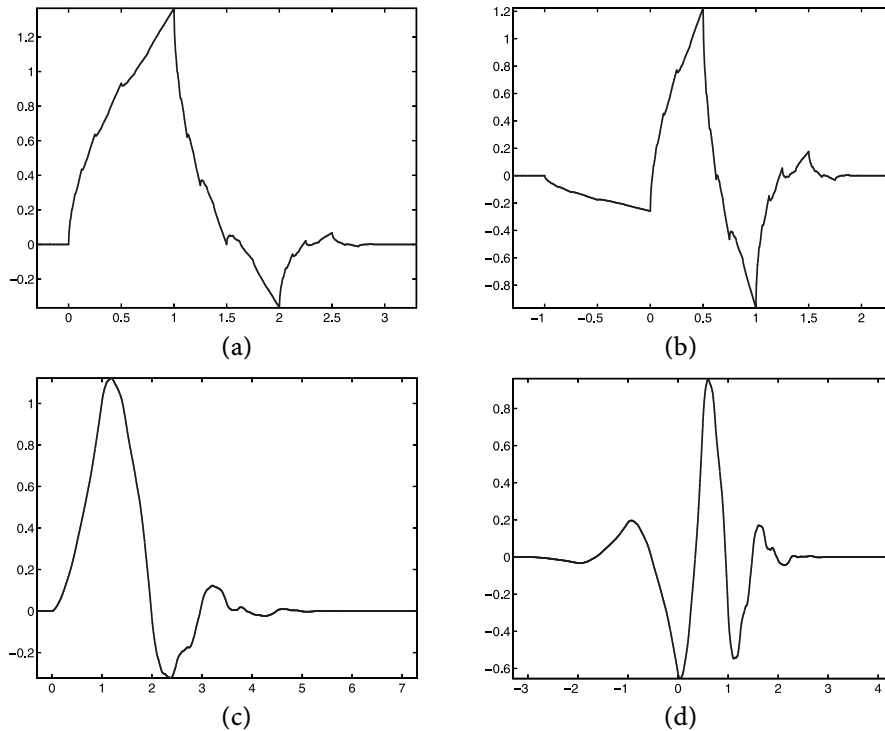


Figure 7.9. Wavelet functions from Daubechies family. (a) Daubechies scaling function, 2 vanishing moments, 4 tap filter (b) Wavelet function corresponding to (a), (c) Daubechies scaling function, 4 vanishing moments, 8 tap filter (d) Wavelet function corresponding to (c)

7.4

Discrete Wavelet Transforms

Discrete wavelet transforms (DWT) are applied to discrete data sets and produce discrete outputs. Transforming signals and data vectors by DWT is a process that resembles the fast Fourier transform (FFT), the Fourier method applied to a set of discrete measurements.

The analogy between Fourier and wavelet methods is even more complete (Table 7.2) when we take into account the continuous wavelet transform and wavelet series expansions.

Discrete wavelet transforms map data from the time domain (the original or input data vector) to the wavelet domain. The result is a vector of the same size. Wavelet transforms are linear and they can be defined by matrices of dimension

Table 7.2. The analogy between Fourier and wavelet methods

Fourier Methods	Fourier Integrals	Fourier Series	Discrete Fourier Transforms
Wavelet Methods	Continuous Wavelet Transforms	Wavelet Series	Discrete Wavelet Transforms

$n \times n$ if they are applied to inputs of size n . Depending on boundary conditions, such matrices can be either orthogonal or “close” to orthogonal. When the matrix is orthogonal, the corresponding transform is a rotation in \mathbb{R}^n in which the data (a n -tuple) is a point in \mathbb{R}^n . The coordinates of the point in the rotated space comprise the discrete wavelet transform of the original coordinates. Here we provide two toy examples.

Example 5 Let the vector be $(1, 2)$ and let $M(1, 2)$ be the point in \mathbb{R}^2 with coordinates given by the data vector. The rotation of the coordinate axes by an angle of $\pi/4$ can be interpreted as a DWT in the Haar wavelet basis. The rotation matrix is

5

$$W = \begin{pmatrix} \cos \frac{\pi}{4} & \sin \frac{\pi}{4} \\ \sin \frac{\pi}{4} & \cos \frac{\pi}{4} \end{pmatrix} = \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{pmatrix},$$

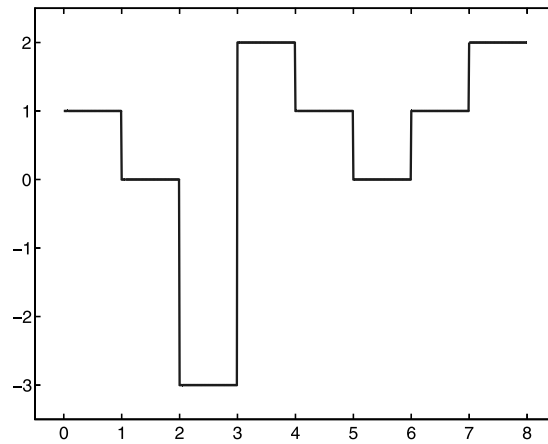
and the discrete wavelet transform of $(1, 2)'$ is $W \cdot (1, 2)' = (3/\sqrt{2}, -1/\sqrt{2})'$. Notice that *the energy* (squared distance of the point from the origin) is preserved, $1^2 + 2^2 = (1/\sqrt{2})^2 + (3/\sqrt{2})^2$, since W is a rotation.

Example 6 Let $y = (1, 0, -3, 2, 1, 0, 1, 2)$. The associated function f is given in Fig. 7.10. The values $f(n) = y_n, n = 0, 1, \dots, 7$ are interpolated by a piecewise constant function. We assume that f belongs to Haar’s multiresolution space V_0 .

6

The following matrix equation gives the connection between y and the wavelet coefficients (data in the wavelet domain).

$$\begin{bmatrix} 1 \\ 0 \\ -3 \\ 2 \\ 1 \\ 0 \\ 1 \\ 2 \end{bmatrix} = \begin{bmatrix} \frac{1}{2\sqrt{2}} & \frac{1}{2\sqrt{2}} & \frac{1}{2} & 0 & \frac{1}{\sqrt{2}} & 0 & 0 & 0 \\ \frac{1}{2\sqrt{2}} & \frac{1}{2\sqrt{2}} & \frac{1}{2} & 0 & -\frac{1}{\sqrt{2}} & 0 & 0 & 0 \\ \frac{1}{2\sqrt{2}} & \frac{1}{2\sqrt{2}} & -\frac{1}{2} & 0 & 0 & \frac{1}{\sqrt{2}} & 0 & 0 \\ \frac{1}{2\sqrt{2}} & \frac{1}{2\sqrt{2}} & -\frac{1}{2} & 0 & 0 & -\frac{1}{\sqrt{2}} & 0 & 0 \\ \frac{1}{2\sqrt{2}} & -\frac{1}{2\sqrt{2}} & 0 & \frac{1}{2} & 0 & 0 & \frac{1}{\sqrt{2}} & 0 \\ \frac{1}{2\sqrt{2}} & -\frac{1}{2\sqrt{2}} & 0 & \frac{1}{2} & 0 & 0 & -\frac{1}{\sqrt{2}} & 0 \\ \frac{1}{2\sqrt{2}} & -\frac{1}{2\sqrt{2}} & 0 & -\frac{1}{2} & 0 & 0 & 0 & \frac{1}{\sqrt{2}} \\ \frac{1}{2\sqrt{2}} & -\frac{1}{2\sqrt{2}} & 0 & -\frac{1}{2} & 0 & 0 & 0 & -\frac{1}{\sqrt{2}} \end{bmatrix} \cdot \begin{bmatrix} c_{00} \\ d_{00} \\ d_{10} \\ d_{11} \\ d_{20} \\ d_{21} \\ d_{22} \\ d_{23} \end{bmatrix}.$$

Figure 7.10. A function interpolating y on $[0, 8)$

The solution is

$$\begin{bmatrix} c_{00} \\ d_{00} \\ d_{10} \\ d_{11} \\ d_{20} \\ d_{21} \\ d_{22} \\ d_{23} \end{bmatrix} = \begin{bmatrix} \sqrt{2} \\ -\sqrt{2} \\ 1 \\ -1 \\ \frac{1}{\sqrt{2}} \\ -\frac{5}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} \end{bmatrix}.$$

Thus,

$$\begin{aligned} f = & \sqrt{2}\phi_{-3,0} - \sqrt{2}\psi_{-3,0} + \psi_{-2,0} - \psi_{-2,1} \\ & + \frac{1}{\sqrt{2}}\psi_{-1,0} - \frac{5}{\sqrt{2}}\psi_{-1,1} + \frac{1}{\sqrt{2}}\psi_{-1,2} - \frac{1}{\sqrt{2}}\psi_{-1,3}. \end{aligned} \quad (7.45)$$

The solution is easy to verify. For example, when $x \in [0, 1)$,

$$f(x) = \sqrt{2} \cdot \frac{1}{2\sqrt{2}} - \sqrt{2} \cdot \frac{1}{2\sqrt{2}} + 1 \cdot \frac{1}{2} + \frac{1}{\sqrt{2}} \cdot \frac{1}{\sqrt{2}} = \frac{1}{2} + \frac{1}{2} = 1 (= y_0).$$

Applying wavelet transforms by multiplying the input vector with an appropriate orthogonal matrix is conceptually straightforward task, but of limited practical value. Storing and manipulating the transformation matrices for long inputs ($n > 2000$) may not even be feasible.

This obstacle is solved by the link of discrete wavelet transforms with fast filtering algorithms from the field of signal and image processing.

The Cascade Algorithm

7.4.1

Mallat (1989a,b) was the first to link wavelets, multiresolution analyses and cascade algorithms in a formal way. Mallat's cascade algorithm gives a constructive and efficient recipe for performing the discrete wavelet transform. It relates the wavelet coefficients from different levels in the transform by filtering with wavelet filter h and its mirror counterpart g .

It is convenient to link the original data with the space V_J , where J is often 0 or $\log n$, where n is a dyadic size of data. Then, coarser smooth and complementing detail spaces are (V_{J-1}, W_{J-1}) , (V_{J-2}, W_{J-2}) , etc. Decreasing the index in V -spaces is equivalent to coarsening the approximation to the data.

By a straightforward substitution of indices in the scaling equations (7.21) and (7.35), one obtains

$$\phi_{j-1,l}(x) = \sum_{k \in \mathbb{Z}} h_{k-2l} \phi_{jk}(x) \quad \text{and} \quad \psi_{j-1,l}(x) = \sum_{k \in \mathbb{Z}} g_{k-2l} \phi_{jk}(x). \quad (7.46)$$

The relations in (7.46) are fundamental in developing the cascade algorithm.

In a multiresolution analysis, $\dots \subset V_{j-1} \subset V_j \subset V_{j+1} \subset \dots$. Since $V_j = V_{j-1} \oplus W_{j-1}$, any function $v_j \in V_j$ can be represented uniquely as $v_j(x) = v_{j-1}(x) + w_{j-1}(x)$, where $v_{j-1} \in V_{j-1}$ and $w_{j-1} \in W_{j-1}$. It is customary to denote the coefficients associated with $\phi_{jk}(x)$ and $\psi_{jk}(x)$ by c_{jk} and d_{jk} , respectively.

Thus,

$$\begin{aligned} v_j(x) &= \sum_k c_{j,k} \phi_{j,k}(x) \\ &= \sum_l c_{j-1,l} \phi_{j-1,l}(x) + \sum_l d_{j-1,l} \psi_{j-1,l}(x) \\ &= v_{j-1}(x) + w_{j-1}(x). \end{aligned}$$

By using the general scaling equations (7.46), orthogonality of $w_{j-1}(x)$ and $\phi_{j-1,l}(x)$ for any j and l , and additivity of inner products, we obtain

$$\begin{aligned} c_{j-1,l} &= \langle v_j, \phi_{j-1,l} \rangle \\ &= \left\langle v_j, \sum_k h_{k-2l} \phi_{j,k} \right\rangle \\ &= \sum_k h_{k-2l} \langle v_j, \phi_{j,k} \rangle \\ &= \sum_k h_{k-2l} c_{j,k}. \end{aligned} \quad (7.47)$$

Similarly $d_{j-1,l} = \sum_k g_{k-2l} c_{j,k}$.

The cascade algorithm works in the reverse direction as well. Coefficients in the next finer scale corresponding to V_j can be obtained from the coefficients corresponding to V_{j-1} and W_{j-1} . The relation

$$\begin{aligned} c_{j,k} &= \langle v_j, \phi_{j,k} \rangle \\ &= \sum_l c_{j-1,l} \langle \phi_{j-1,l}, \phi_{j,k} \rangle + \sum_l d_{j-1,l} \langle \psi_{j-1,l}, \phi_{j,k} \rangle \\ &= \sum_l c_{j-1,l} h_{k-2l} + \sum_l d_{j-1,l} g_{k-2l}, \end{aligned} \quad (7.48)$$

describes a single step in the reconstruction algorithm.

The discrete wavelet transform can be described in terms of operators. Let the operators \mathcal{H} and \mathcal{G} acting on a sequence $a = \{a_n, n \in \mathbb{Z}\}$, satisfy the following coordinate-wise relations:

$$(\mathcal{H}a)_k = \sum_n h_{n-2k} a_n \quad (\mathcal{G}a)_k = \sum_n g_{n-2k} a_n,$$

and their adjoint operators \mathcal{H}^* and \mathcal{G}^* satisfy:

$$(\mathcal{H}^*a)_n = \sum_k h_{n-2k} a_k \quad (\mathcal{G}^*a)_n = \sum_k g_{n-2k} a_k,$$

where $\mathbf{h} = \{h_n\}$ is wavelet filter and $\mathbf{g} = \{g_n\}$ its quadrature-mirror counterpart.

Denote the original signal by $\mathbf{c}^{(J)} = \{c_k^{(J)}\}$. If the signal is of length 2^J , then $\mathbf{c}^{(J)}$ can be interpolated by the function $f(x) = \sum_k c_k^{(J)} \phi(x-k)$ from V_J . In each step of the wavelet transform, we move to the next coarser approximation (level) $\mathbf{c}^{(j-1)}$ by applying the operator \mathcal{H} , $\mathbf{c}^{(j-1)} = \mathcal{H}\mathbf{c}^{(j)}$. The “detail information,” lost by approximating $\mathbf{c}^{(j)}$ by the “averaged” $\mathbf{c}^{(j-1)}$, is contained in vector $\mathbf{d}^{(j-1)} = \mathcal{G}\mathbf{c}^{(j)}$.

The discrete wavelet transform of a sequence $\mathbf{y} = \mathbf{c}^{(J)}$ of length 2^J can then be represented as

$$\left(\mathbf{c}^{(J-k)}, \mathbf{d}^{(J-k)}, \mathbf{d}^{(J-k+1)}, \dots, \mathbf{d}^{(J-2)}, \mathbf{d}^{(J-1)} \right). \quad (7.49)$$

Notice that the lengths of \mathbf{y} and its transform in (7.49) coincide. Because of decimation, the length of $\mathbf{c}^{(j)}$ is twice the length of $\mathbf{c}^{(j-1)}$, and $2^J = 2^{J-k} + \sum_{i=1}^k 2^{J-i}$, $1 \leq k \leq J$.

For an illustration of (7.49), see Fig. 7.11. By utilizing the operator notation, it is possible to summarize the discrete wavelet transform (curtailed at level k) in a single line:

$$\mathbf{y} \mapsto (\mathcal{H}^k \mathbf{y}, \mathcal{G} \mathcal{H}^{k-1} \mathbf{y}, \dots, \mathcal{G} \mathcal{H}^2 \mathbf{y}, \mathcal{G} \mathcal{H} \mathbf{y}, \mathcal{G} \mathbf{y}).$$

The number k can be any arbitrary integer between 1 and J and it is associated with the coarsest “smooth” space, V_{J-k} , up to which the transform was curtailed. In terms of multiresolution spaces, (7.49) corresponds to the multiresolution de-

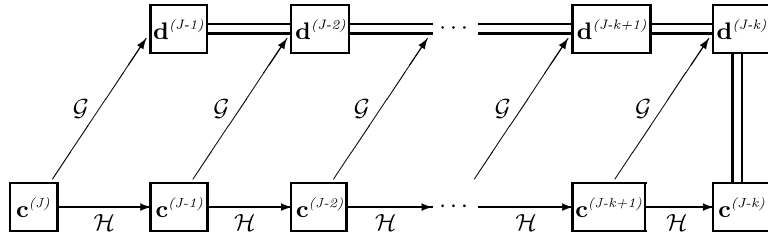


Figure 7.11. Forward wavelet transform of depth k (DWT is a vector of coefficients connected by double lines)

composition $V_{J-k} \oplus W_{J-k} \oplus W_{J-k+1} \oplus \dots \oplus W_{J-1}$. When $k = J$ the vector $c^{(0)}$ contains a single element, $c^{(0)}$.

If the wavelet filter length exceeds 2, one needs to define actions of the filter beyond the boundaries of the sequence to which the filter is applied. Different policies are possible. The most common is a periodic extension of the original signal.

The reconstruction formula is also simple in terms of operators \mathcal{H}^* and \mathcal{G}^* . They are applied on $c^{(j-1)}$ and $d^{(j-1)}$, respectively, and the results are added. The vector $c^{(j)}$ is reconstructed as

$$c^{(j)} = \mathcal{H}^* c^{(j-1)} + \mathcal{G}^* d^{(j-1)}, \tag{7.50}$$

Recursive application of (7.50) leads to

$$\begin{aligned} & (\mathcal{H}^k y, \mathcal{G} \mathcal{H}^{k-1} y, \dots, \mathcal{G} \mathcal{H}^2 y, \mathcal{G} \mathcal{H} y, \mathcal{G} y) \\ &= (c^{(J-k)}, d^{(J-k)}, d^{(J-k+1)}, \dots, d^{(J-2)}, d^{(J-1)}) \\ &\mapsto \sum_{i=1}^{k-1} (\mathcal{H}^*)^{k-1-i} \mathcal{G}^* d^{(J-k+i)} + (\mathcal{H}^*)^k c^{(J-k)} = y. \end{aligned}$$

Example 7 Let $y = (1, 0, -3, 2, 1, 0, 1, 2)$ be an exemplary set we want to transform by Haar's DWT. Let $k = J = 3$, i.e., the coarsest approximation and detail levels

7

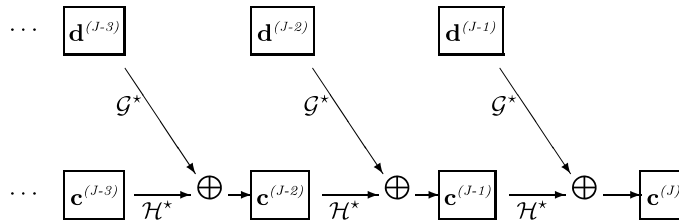


Figure 7.12. Inverse Transform

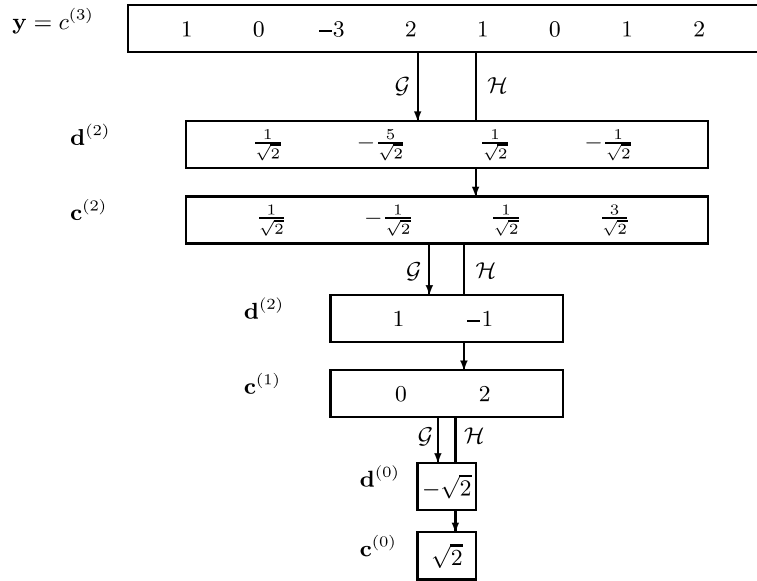


Figure 7.13. An illustration of a decomposition procedure

will contain a single point each. The decomposition algorithm applied on $y = (1, 0, -3, 2, 1, 0, 1, 2)$ is given schematically in Fig. 7.13.

For the Haar wavelet, the operators \mathcal{H} and \mathcal{G} are given by $(\mathcal{H}a)_k = \sum_n h_{n-2k}a_n = \sum_m h_m a_{m+2k} = h_0 a_{2k} + h_1 a_{2k+1} = (a_{2k} + a_{2k+1})/\sqrt{2}$. Similarly, $(\mathcal{G}a)_k = \sum_n g_{n-2k}a_n = \sum_m g_m a_{m+2k} = g_0 a_{2k} + g_1 a_{2k+1} = (a_{2k} - a_{2k+1})/\sqrt{2}$.

The reconstruction algorithm is given in Fig. 7.14. In the process of reconstruction, $(\mathcal{H}^*a)_n = \sum_k h_{n-2k}a_k$, and $(\mathcal{G}^*a)_n = \sum_k g_{n-2k}a_k$. For instance, the first line in Fig. 7.14 recovers the object $\{1, 1\}$ from $\sqrt{2}$ by applying \mathcal{H}^* . Indeed, $(\mathcal{H}^*\{a_0\})_0 = h_0\sqrt{2} = 1$ and $(\mathcal{H}^*\{a_0\})_1 = h_1\sqrt{2} = 1$.

We already mentioned that when the length of the filter exceeds 2, boundary problems occur since the convolution goes outside the range of data.

There are several approaches to resolving the boundary problem. The signal may be continued in a periodic way $(\dots, y_{n-1}, y_n | y_1, y_2, \dots)$, symmetric way $(\dots, y_{n-1}, y_n | y_{n-1}, y_{n-2}, \dots)$, padded by a constant, or extrapolated as a polynomial. Wavelet transforms can be confined to an interval (in the sense of Cohen, Daubechies and Vial (1993) and periodic and symmetric extensions can be viewed as special cases. Periodized wavelet

transforms are also defined in a simple way.

If the length of the data set is not a power of 2, but of the form $M \cdot 2^K$, for M odd and K a positive integer, then only K steps in the decomposition algorithm can be performed. For precise descriptions of conceptual and calculational hurdles

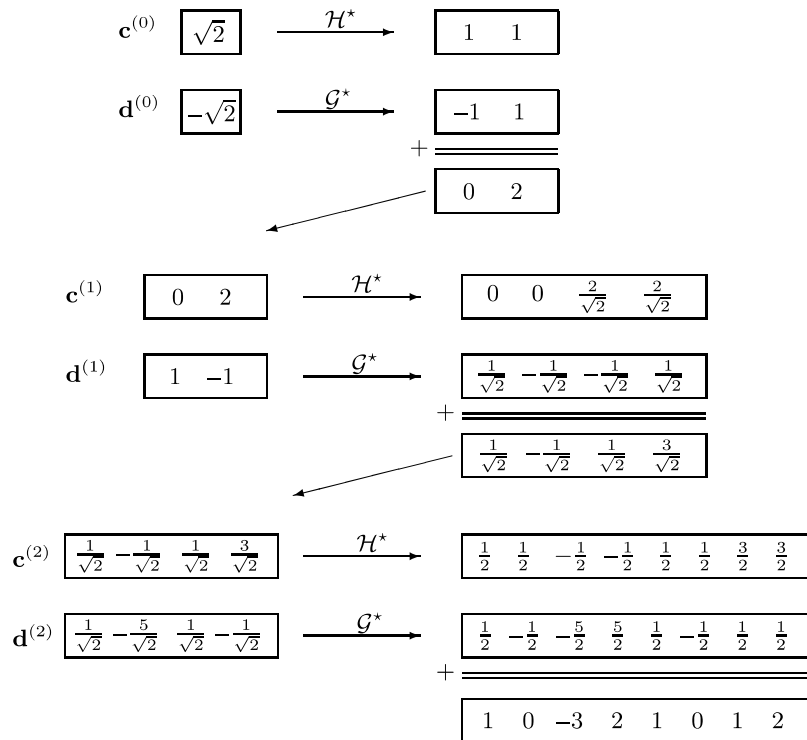


Figure 7.14. An illustration of a reconstruction procedure

caused by boundaries and data sets whose lengths are not a power of 2, we direct the reader to the monograph by Wickerhauser (1994).

In this section we discussed the most basic wavelet transform. Various generalizations include biorthogonal wavelets, multiwavelets, nonseparable multidimensional wavelet transforms, complex wavelets, lazy wavelets, and many more.

For various statistical applications of wavelets (nonparametric regression, density estimation, time series, deconvolutions, etc.) we direct the reader to Antoniadis (1997), Härdle et al. (1998), Vidakovic (1999). An excellent monograph by Walter and Shen (2000) discusses statistical applications of wavelets and various other orthogonal systems.

Matlab Implementation of Cascade Algorithm

The following two matlab m-files implement discrete wavelet transform and its inverse, with periodic handling of boundaries. The data needs to be of dyadic size (power of 2). The programs are didactic, rather than efficient. For an excellent and comprehensive wavelet package, we direct the reader to wavelab802 module (<http://www-stat.stanford.edu/~wavelab/>) maintained by Donoho and his coauthors.

```

function dwtr = dwtr(data, L, filterh)
% function dwtr = dwt(data, filterh, L);
% Calculates the DWT of periodic data set
% with scaling filter filterh and L scales.
%
% Example of Use:
% data = [1 0 -3 2 1 0 1 2]; filter = [sqrt(2)/2 sqrt(2)/2];
% wt = DWTR(data, 3, filter)
%-----

n = length(filterh);           %Length of wavelet filter
C = data;                      %Data \qut{live} in V_J
dwtr = [];                     %At the beginning dwtr empty
H =fliplr(filterh);           %Flip because of convolution
G = filterh;                  %Make quadrature mirror
G(1:2:n) = -G(1:2:n);         % counterpart
for j = 1:L                    %Start cascade
    nn = length(C);           %Length needed to
    C = [C(mod((-n-1):-1),nn)+1] C]; % make periodic
    D = conv(C,G);            %Convolve,
    D = D([n:2:(n+nn-2)]+1);  % keep periodic, decimate
    C = conv(C,H);            %Convolve,
    C = C([n:2:(n+nn-2)]+1);  % keep periodic, decimate
    dwtr = [D,dwtr];          %Add detail level to dwtr
end;                            %Back to cascade or end
dwtr = [C, dwtr];             %Add the last \qut{smooth} part
function data = idwtr(wtr, L, filterh)
% function data = idwt(wtr, L, filterh);
% Calculates the IDWT of wavelet
% transform wtr using wavelet filter
% \qut{filterh} and L scales.
% Example:
%>> max(abs(data - IDWTR(DWTR(data,3,filter), 3,filter)))
%ans = 4.4409e-016
%-----
nn = length(wtr);   n = length(filterh); %Lengths
if nargin==2, L = round(log2(nn)); end; %Depth of transform
H = filterh;       %Wavelet H filter
G =fliplr(H); G(2:2:n) = -G(2:2:n); %Wavelet G filter
LL = nn/(2^L);    %Number of scaling coeffs
C = wtr(1:LL);   %Scaling coeffs
for j = 1:L      %Cascade algorithm
    w = mod(0:n/2-1,LL)+1; %Make periodic
    D = wtr(LL+1:2*LL); %Wavelet coeffs
    Cu(1:2:2*LL+n) = [C C(1,w)]; %Upsample & keep periodic
    Du(1:2:2*LL+n) = [D D(1,w)]; %Upsample & keep periodic
    C = conv(Cu,H) + conv(Du,G); %Convolve & add
    C = C([n:n+2*LL-1]-1); %Periodic part
    LL = 2*LL; %Double the size of level
end;
data = C; %The inverse DWT

```

7.5

Conclusion

In this chapter we gave an overview of several transforms useful in computational statistics. We emphasized frequency and scale domain transforms (Fourier and wavelet) since they provide an insight to the phenomena, not available in the domain of untransformed data. Moreover, multiscale transforms are relatively new, and as such deserve more attention. It was pretentious to title this chapter *Transforms in Statistics*, since literally several dozens important transforms are not

even mentioned. As it was hinted in the introduction, a just task of over-viewing all important transformations used in statistical practice would take a space of a large monograph.

Acknowledgements. Work on this chapter was supported by DOD/NSA Grant E-24-60R at Georgia Institute of Technology. Editor Jim Gentle read early versions of the chapter and gave many valuable comments. All `matlab` programs that produced figures and simulations are available from the author at request.

References

- Anderson, T.W. (1984). An Introduction to Multivariate Statistical Analysis, Second Edition, Wiley, New York.
- Antoniadis, A. (1997). Wavelets in statistics: A Review. *J. Ital. Statist. Soc.*, 6: 97–144.
- Baraniuk, R.G. (1994). Wigner–Ville spectrum estimation via wavelet soft-thresholding. In *Proc. IEEE-SP Int. Symp. on Time-Frequency and Time-Scale Analysis*, Philadelphia.
- Box, G.E.P. and Cox, D.R. (1964). An Analysis of Transformations, *Journal of the Royal Statistical Society*, 26: 211–243, discussion 244–252.
- Brigham, E.O. (1988). *The Fast Fourier Transform and Its Applications*, Prentice-Hall, Englewood Cliffs, NJ.
- Carmona, R., Hwang, W-L. and Torr sani, B. (1998). *Practical Time-Frequency Analysis*, volume 9 of *Wavelet Analysis and its Applications*, Academic Press, San Diego.
- Cohen, A, Daubechies, I. and Vial, P. (1993). Wavelets on the interval and fast wavelet transforms. *Appl. Comput. Harmon. Anal.*, 1(1): 54–81.
- Daubechies, I. (1992). *Ten Lectures on Wavelets*, Number 61 in CBMS-NSF Series in Applied Mathematics, SIAM, Philadelphia.
- Feuerverger, A. and Mureika, R. (1977). The empirical characteristic function and its applications, *The Annals of Statistics*, 5: 88–97.
- Flandrin, P. (1992). Time-scale analyses and self-similar stochastic processes. In Byrnes et al. (eds), *Wavelets and Their Applications*, pp. 121–142, NATO ASI Series vol. 442.
- Flandrin, P. (1999). *Time-Frequency/Time-scale Analysis*, Academic Press, 386pp.
- Gabor, D. (1946). Theory of communication. *J. IEEE*, 93: 429–457.
- Grossmann, A. and Morlet, J. (1984). Decomposition of Hardy functions into square integrable wavelets of constant shape. *SIAM J. Math.*, 15: 723–736.
- Grossmann, A. and Morlet, J. (1985). Decomposition of functions into wavelets of constant shape and related transforms. In Streit, L. (ed), *Mathematics and physics, lectures on recent results*, World Scientific, River Edge, NJ.
- H rdle, W., Kerkycharian, G., Pickard, D. and Tsybakov, A. (1998). *Wavelets, Approximation, and Statistical Applications*, Lecture Notes in Statistics 129. Springer-Verlag, New York.

- Mallat, S.G. (1989a). Multiresolution approximations and wavelet orthonormal bases of $L^2(\mathbb{R})$. *Trans. Amer. Math. Soc.*, 315: 69–87.
- Mallat, S.G. (1989b). A theory for multiresolution signal decomposition: The wavelet representation. *IEEE Trans. on Patt. Anal. Mach. Intell.*, 11(7): 674–693.
- Mallat, S.G. (1999). *A Wavelet Tour of Signal Processing*, Second Edition. Academic Press, San Diego.
- Morlet, J., Arens, G., Fourgeau, E. and Giard, D. (1982). Wave propagation and sampling theory. *Geophys.*, 47: 203–236.
- Murata, N. (2001). Properties of the empirical characteristic function and its application to testing for independence. In Lee, Jung, Makeig, and Sejnowski (eds), *Proceedings ICA2001, 3rd International Conference on Independent Component Analysis*, San Diego.
- Pensky, M. and Vidakovic, B. (2003) Bayesian decision theoretic scale-adaptive estimation of log-spectral density. Technical Report 01-2003, ISyE, Georgia Institute of Technology. <http://www.isye.gatech.edu/~brani/isyestat/>.
- Tong, H. (1996). *Non-Linear Time Series*, Clarendon Press, Oxford.
- Vidakovic, B. (1999). *Statistical Modeling by Wavelets*, Wiley, NY.
- Ville, J. (1948). Théorie et Applications de la Notion de Signal Analytique, *Cables et Transmission*, 2A: 61–74.
- Walter, G.G. and Shen, X. (2000). *Wavelets and Other Orthogonal Systems*, Second Edition, CRC Press.
- Wickerhauser, M. V. (1994). *Adapted Wavelet Analysis from Theory to Software*, A K Peters, Ltd., Wellesley, MA.

Parallel Computing Techniques **II.8**

Junji Nakano

8.1	<i>Introduction</i>	238
8.2	<i>Basic Ideas</i>	239
	Memory Architectures of Parallel Computers	239
	Costs for Parallel Computing	242
8.3	<i>Parallel Computing Software</i>	244
	Process Forking	245
	Threading	247
	OpenMP	251
	PVM	253
	MPI.....	256
	HPF	260
8.4	<i>Parallel Computing in Statistics</i>	262
	Parallel Applications in Statistical Computing	262
	Parallel Software for Statistics	263

Introduction

Parallel computing means to divide a job into several tasks and use more than one processor simultaneously to perform these tasks. Assume you have developed a new estimation method for the parameters of a complicated statistical model. After you prove the asymptotic characteristics of the method (for instance, asymptotic distribution of the estimator), you wish to perform many simulations to assure the goodness of the method for reasonable numbers of data values and for different values of parameters. You must generate simulated data, for example, 100,000 times for each length and parameter value. The total simulation work requires a huge number of random number generations and takes a long time on your PC. If you use 100 PCs in your institute to run these simulations simultaneously, you may expect that the total execution time will be $1/100$. This is the simple idea of parallel computing.

Computer scientists noticed the importance of parallel computing many years ago (Flynn, 1966). It is true that the recent development of computer hardware has been very rapid. Over roughly 40 years from 1961, the so called “Moore’s law” holds: the number of transistors per silicon chip has doubled approximately every 18 months (Tuomi, 2002). This means that the capacity of memory chips and processor speeds have also increased roughly exponentially. In addition, hard disk capacity has increased dramatically. Consequently, modern personal computers are more powerful than “super computers” were a decade ago. Unfortunately, even such powerful personal computers are not sufficient for our requirements. In statistical analysis, for example, while computers are becoming more powerful, data volumes are becoming larger and statistical techniques are becoming more computer intensive. We are continuously forced to realize more powerful computing environments for statistical analysis. Parallel computing is thought to be the most promising technique.

However, parallel computing has not been popular among statisticians until recently (Schervish, 1988). One reason is that parallel computing was available only on very expensive computers, which were installed at some computer centers in universities or research institutes. Few statisticians could use these systems easily. Further, software for parallel computing was not well prepared for general use.

Recently, cheap and powerful personal computers changed this situation. The Beowulf project (Sterling et al., 1999), which realized a powerful computer system by using many PCs connected by a network, was a milestone in parallel computer development. Freely available software products for parallel computing have become more mature. Thus, parallel computing has now become easy for statisticians to access.

In this chapter, we describe an overview of available technologies for parallel computing and give examples of their use in statistics. The next section considers the basic ideas of parallel computing, including memory architectures. Section 8.3 introduces the available software technologies such as process forking, threading, OpenMP, PVM (Parallel Virtual Machine), MPI (Message Passing Interface) and HPF (High Performance Fortran). The last section describes some examples of parallel computing in statistics.

Basic Ideas

8.2

Two important parts of computer hardware are the processor, which performs computations, and memory, in which programs and data are stored. A processor is also often called a central processing unit (CPU). Modern computer systems adopt a stored programming architecture: all the program instructions are stored in memory together with processed data and are executed sequentially by a processor according to the instructions.

In a traditional single processor computer, a single stream of instructions is generated from the program, and these instructions operate on a single stream of data. Flynn (1966) called this arrangement a single instruction stream–single data stream (SISD) computer.

On the other hand, a parallel computer system uses several processors, and is realized as a single instruction stream–multiple data stream (SIMD) computer or a multiple instruction stream–multiple data stream (MIMD) computer. SIMD refers to a form of parallel execution in which all processors execute the same operation on different data at the same time, and is often associated with performing the same operation on every element of a vector or array. MIMD refers to parallel execution in which each processor works independently; for example, one processor might update a database file while another processor handles a graphic display.

The fundamental software of a modern computer system is an operating system such as UNIX or Microsoft Windows. They support multiple users and multiple tasks, even on single processor systems, by adopting time-slicing mechanisms, in which a processor executes tasks cyclically. In parallel computer systems, some tasks are executed on different processors simultaneously.

Memory Architectures of Parallel Computers

8.2.1

The traditional computer system has a single processor (or CPU) that can access all of the memory (Fig. 8.1). Parallel computers use more than one processor simultaneously for a single calculation task. There are two simple methods to increase the number of available processors in a single system. One method is to add processors to the traditional single processor system without changing other parts. Because all the memory is shared by all processors, such systems are called shared memory systems (Fig. 8.2). An example of a shared memory system is a dual processor personal computer, where the motherboard has two sockets for CPUs. When we mount one CPU, it works as a traditional single processor system. If we mount two CPUs, both processors can access all the memory in the PC, and it works as a shared memory system. A second method is to connect traditional single processor computers by a network. This is called a distributed memory system, because the memory is used by a single processor locally and is “distributed” over the whole system (Fig. 8.3). An example of a distributed memory system is a network of workstations, in which each node computer works independently and communicates with the others through a network to solve a single problem.



Figure 8.1. Traditional system

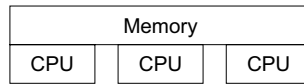


Figure 8.2. Shared memory system

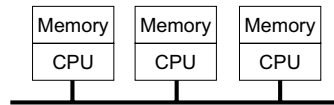


Figure 8.3. Distributed memory system

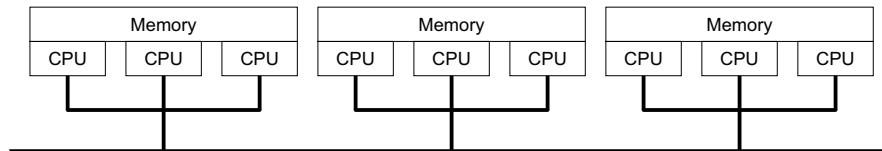


Figure 8.4. Distributed shared memory system

Integration of shared memory and distributed memory is possible (Fig. 8.4). Network-connected PCs that each have two processors can be considered a distributed shared memory system.

Shared Memory Systems

In the simple shared memory realization, all the processors can access all the memory at the same speed by using a common memory bus. This is known as a uniform memory access (UMA) configuration. Performance in a UMA system is limited by the memory bus bandwidth; adding processors to the system beyond some point does not increase performance linearly, because signals from processors flow on the same memory bus and often cause collisions. Typically, UMA configurations do not scale well beyond 10 to 20 processors.

To improve communication between processors and memory, a non-uniform memory access (NUMA) configuration is used. In NUMA systems, all processors have access to all the memory, but the cost of accessing a specific location in memory is different for different processors, because different regions of memory are on physically different buses. Even if we adopt a NUMA configuration, it is not efficient to use more than 100 processors in a shared memory system.

A shared memory system is also a symmetric multiprocessor (SMP) system, in which any processor can do equally well any piece of work.

In a shared memory system, a single copy of an operating system is in charge of all the processors and the memory. It usually uses a programming model called

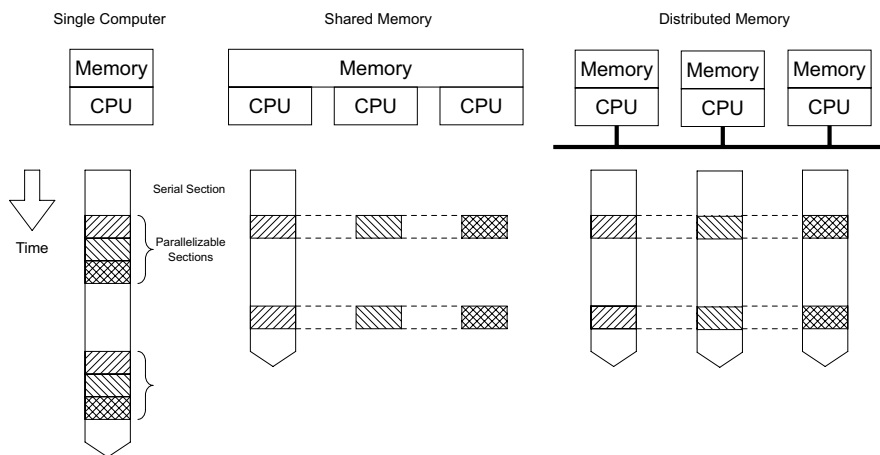


Figure 8.5. Typical parallel computing execution

“fork–join”. Each program begins by executing just a single task, called the master. When the first parallel work is reached, the master spawns (or forks) additional tasks (called slaves or workers), which will “join” to the master when they finish their work (the middle figure in Fig. 8.5). Such activities can be programmed by using software technologies such as process, thread or OpenMP, which will be explained in the next section.

Distributed Memory Systems

In a distributed memory system, each node computer is an independent computer that has, at least, processor and memory, and the nodes are connected together by a network. This so called “network of workstations” (NOW) is the cheapest way to construct a distributed memory system, because we can utilize many different kinds of workstations available, connected by a network, without adding any new hardware. However, NOW is sometimes ineffective for heavy computation, because, for example, general purpose networks are slow, and nodes may be unexpectedly used for other work, so that it is difficult to schedule them efficiently.

Nowadays, “Beowulf class cluster computers” are popular for distributed memory parallel computing (Sterling et al., 1999). These are a kind of NOW, but there are slight differences. First, the nodes in the cluster are the same kind of workstation or PC, and are dedicated to the cluster calculation tasks. Typically, node computers share the working directory on the hard disk and have no display or keyboard. The interconnection network is isolated from external networks and is also dedicated to the cluster, and communication among the nodes can be done without further authentication. Operating system parameters are tuned to improve the total performance for parallel computing. All these characteristics help the performance of the parallel computing on the cluster.

Distributed memory systems have no memory bus problem. Each processor can use the full bandwidth to its own local memory without interference from other processors. Thus, there is no inherent limit to the number of processors. The size of the system is constrained only by the network used to connect the node computers. Some distributed memory systems consist of several thousand processors.

As nodes in a distributed memory system share no memory at all, exchange of information among processors is more difficult than in a shared memory system. We usually adopt a message passing programming model on a distributed memory system; we organize a program as a set of independent tasks that communicate with each other via messages. This introduces two sources of overhead: it takes time to construct and send a message from one processor to another, and the receiving processor must be interrupted to deal with messages from other processors.

Available message passing libraries are PVM and MPI. The right figure in Fig. 8.5 shows an execution image of MPI. HPF is also mainly used in distributed memory systems. These libraries are illustrated in the next section.

8.2.2 Costs for Parallel Computing

We expect that the calculation speed increases n times if we use n processors instead of one. We also wish to use multiprocessor systems just like an ordinary single processor system. However, some costs are incurred in realizing parallel computing. They include the non-parallel characteristics of the problem, communication costs such as distributing and gathering data and/or programs, the difficulty of programming for synchronization among executions and unexpected influences of cache memory. All these factors reduce the effect of parallelization.

Amdahl's Law

All programming tasks include non-parallelizable or serial parts, which cannot be executed on several processors, for example, summarizing calculation results and writing them to the display or file. Assume the ratio of computing time for the serial parts to the whole task is f ($0 < f < 1$). If a single processor requires t_s time to complete the task, $(1-f)t_s$ computation time is used for the parallelizable task and ft_s computation time is used for the serial task. If we use n processors, the elapsed time for execution of the parallelizable task will be at least $(1-f)t_s/n$, while the execution time of the serial task remains ft_s . Thus, the ratio of execution time for n processors to that for one processor, $S(n)$, which is called the speedup factor, is

$$S(n) = \frac{t_s}{ft_s + (1-f)t_s/n} = \frac{n}{1 + (n-1)f}.$$

This equation is known as "Amdahl's law" (Amdahl, 1967). When n is large, it converges to $1/f$, that is, the effect of parallel computing is limited. For example, if $f = 5\%$, the maximum possible speedup is 20, even if we use an infinite number of processors. This may discourage the use of parallel computing.

Of course, as f goes to zero, $S(n)$ converges to n , which is an ideal situation.

Gustafson's Law

Amdahl's law considers the situation where the task size is fixed and the number of processors increases. In real problems, however, we wish to perform larger tasks when the number of processors increases. For example, assume time s is required for preparing a task, and time p is required for the (moderate) simulation task. When a parallel computer is available, we wish to perform more simulations, typically, n times larger simulations than the original ones by n processors. To perform this simulation, a single processor system requires $s + np$ time, while the n -processor system requires $s + p$ time. The speedup factor is

$$S(n) = \frac{s + np}{s + p}.$$

This equation is called "Gustafson's law" (Gustafson, 1988). Note that if we define $f = s/(s + np)$, this is as same as Amdahl's law. However, when n becomes large, $S(n)$ becomes large linearly. This means that parallel computing is useful for large-scale problems in which the serial part does not increase as the problem size increases. If s approaches zero, $S(n)$ converges to n , the ideal situation.

Other Costs

If we divide one task into several small tasks and execute them in parallel, we must wait until all the child tasks have been completed: we must synchronize executions. As the slowest child task determines the total execution time, child tasks should be designed to have almost the same execution times, otherwise some processors may be idle while others have tasks queuing for execution. Techniques that aim to spread tasks among the processors equally are called load balancing and are not easy.

In a shared memory system, exchange of information among processors is performed by variables stored in the shared memory. If several tasks use one variable almost simultaneously, it may cause trouble. Consider two tasks trying to decrease the value of variable x by one. Assume $x = 3$; task 1 obtains this value, decreases it and writes 2 into x . If task 2 tries to do the same task before task 1 finishes its work, task 2 also obtains the value 3, and writes 2 into x . Thus, the final result is 2, although x should have decreased twice. To avoid such a maloperation, task 2 must wait until task 1 finishes. All parallel computing software can handle this synchronization problem, typically by using a lock-unlock mechanism.

An important hardware aspect of shared memory systems is cache memory. As the advances in main memory technology do not keep up with processor innovations, memory access is very slow compared with processor speed. In order to solve this problem, another layer of memory has been added between a processor and main memory, called the cache. It is a small amount of very fast, expensive memory, that operates close to the speed of the processor. A separate cache controller monitors memory accesses and loads data and instructions in blocks of contiguous locations from memory into the cache. Once the content of memory is stored in the cache, the processor can operate at full speed by using them. Sometimes, the cache contents are different from the necessary ones. In these cases, the processor

is stalled and has to wait while the necessary data is newly loaded from memory into the cache. This mechanism works well in a single processor system.

All processors in a shared memory system have their own caches. Suppose several processors access the same location of memory and copy them into their caches. If one processor changes the value of the memory in that location, other processors should not use the value in their caches. A cache coherence protocol is used to notify this information among caches. A common cache coherence protocol is an invalidate policy; when one copy of memory is altered, the same data in any other cache is invalidated (by resetting a valid bit in the cache). In shared memory systems, cache coherence is done in the hardware and the programmer need not worry about cache coherence. However, it may cause the slowdown of the calculation speed. Note that caches handle blocks of memory. If one processor writes to one part of the block, copies of the whole block in other caches are invalidated though the actual data is not shared. This is known as false sharing and can damage the performance of the cache in a shared memory system. We are sometimes required to write programs considering the amount of the cache memory in a shared memory system to achieve enough performance.

Distributed memory systems require communication among node computers. Such communication is affected by several factors, including network bandwidth, network latency and communication latency. Network bandwidth is the number of bits that can be transmitted in unit time. Network latency is the time to prepare a message for sending it through the network. Communication latency is the total time to send the message, including software overhead and interface delays. Generally, communication is expensive compared with processor work.

If a problem can be divided into small tasks that are completely independent and require no or very little synchronization and communication, the problem is called “embarrassingly parallel”. Clearly, embarrassingly parallel problems are particularly suitable for parallel computing.

8.3 Parallel Computing Software

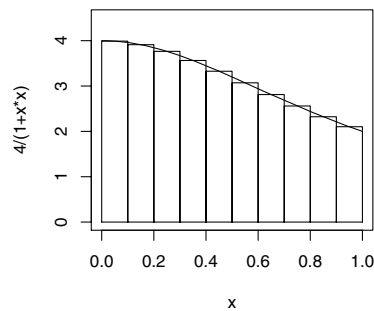
Several well-designed software technologies are available for utilizing parallel computing hardware. Note that each of them is suitable for a specific hardware architecture.

In this section, we use as an example the calculation of the value of π by the approximation formula

$$\pi = \int_0^1 \frac{4}{1+x^2} dx \sim \frac{1}{n} \sum_{i=1}^n \frac{4}{1 + \left(\frac{i-0.5}{n}\right)^2}.$$

The case $n = 10$ is illustrated in Fig. 8.6.

A C program to calculate the last term is given in Listing 1. The main calculation is performed in the `for` statement, which is easily divided into parallel-executed

Figure 8.6. Calculation of π

parts; this is an example of an embarrassingly parallel problem. We show several parallel computing techniques by using this example in this section. We choose this simple example to keep the length of following example source codes as small as possible and to give a rough idea of parallel computing techniques. Note that this example is so simple that only the most fundamental parts of each technique will be used and explained. Many important details of each technique are left to references.

Listing 1

```
#include <stdio.h>

main(int argc, char **argv)
{
    int n, i;
    double d, s, x, pi;
    n = atoi(argv[1]);
    d = 1.0/n;
    s = 0.0;
    for (i=1; i<=n; i++){
        x = (i-0.5)*d;
        s += 4.0/(1.0+x*x);
    }
    pi = d*s;
    printf("pi=%.15f\n", pi);
}
```

Process Forking

8.3.1

Modern operating systems have multi-user and multi-task features even on a single processor; many users can use a single processor system and can seemingly perform many tasks at the same time. This is usually realized by multi-process mechanisms (Tanenbaum, 2001).

UNIX-like operating systems are based on the notion of a process. A process is an entity that executes a given piece of code, has its own execution stack, its own set of memory pages, its own file descriptors table and a unique process ID. Multiprocessing is realized by time-slicing the use of the processor. This technology repeatedly assigns the processor to each process for a short time. As the processor is very fast compared with human activities, it looks as though it is working simultaneously for several users. In shared memory systems, multiprocessing may be performed simultaneously on several processors. Multiprocessing mechanisms are a simple tool for realizing parallel computing.

We can use two processes to calculate the `for` loop in Listing 1, by using the process-handling functions of UNIX operating systems: `fork()`, `wait()` and `exit()`. The function `fork()` creates a new copy process of an existing process. The new process is called the child process, and the original process is called the parent. The return value from `fork()` is used to distinguish the parent from the child; the parent receives the child's process id, but the child receives zero. By using this mechanism, an `if` statement, for example, can be used to prescribe different work for the parent and the child. The child process finishes by calling the `exit()` function, and the parent process waits for the end of the child process by using the `wait()` function. This fork-join mechanism is fundamental to the UNIX operating system, in which the first process to start invokes another process by forking. This procedure is repeated until enough processes are invoked. Although this mechanism was originally developed for one processor and a time-slicing system, UNIX operating systems that support shared memory can run processes on different processors simultaneously.

As processes are independent and share only a limited set of common resources automatically, we must write a program for information exchange among processes. In our example, we use functions to handle shared memory segments: `shmget()`, `shmat()` and `shmctl()`. `shmget()` allocates a shared memory segment, `shmat()` attaches the shared memory segment to the process, and `shmctl()` allows the user to set information such as the owner, group and permissions on the shared memory segment. When the parent process uses `fork()`, the shared memory segment is inherited by the child process and both processes can access it.

Listing 2 shows a two-process version of Listing 1. In the `for` statement, the parent process works for $i = 2, 4, 6, \dots$, while the child process works for $i = 1, 3, 5, \dots$. The child process stores its result to `*shared` and the parent process receives the value and adds it to its own result, then prints the final result.

Listing 2

```
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>
#include <sys/ipc.h>
#include <sys/shm.h>
```



```

main(int argc, char **argv)
{
    int n, i;
    double d, s, x, pi;
    int shmid, iproc;
    pid_t pid;
    double *shared;
    n = atoi(argv[1]);
    d = 1.0/n;
    shmid = shmget(IPC_PRIVATE,
                  sizeof(double), (IPC_CREAT | 0600));
    shared = shmat(shmid, 0, 0);
    shmctl(shmid, IPC_RMID, 0);
    iproc = 0;
    if ((pid = fork()) == -1) {
        fprintf(stderr, "The fork failed!\n");
        exit(0);
    } else {
        if (pid != 0) iproc = 1 ;
    }
    s = 0.0;
    for (i=iproc+1; i<=n; i+=2) {
        x = (i-0.5)*d;
        s += 4.0/(1.0+x*x);
    }
    pi = d*s;
    if (pid == 0) {
        *shared = pi;
        exit(0);
    } else {
        wait(0);
        pi = pi + *shared;
        printf("pi=%.15f\n", pi);
    }
}

```

Forking, however, is not appropriate for parallel computing. Much time and memory is required to duplicate everything in the parent process. Further, a complete copy is not always required, because, for example, the forked child process starts execution at the point of the fork.

Threading

8.3.2

As a process created using the UNIX `fork()` function is expensive in setup time and memory space, it is sometimes called a “heavyweight” process. Often

a partial copy of the process is enough and other parts can be shared. Such copies can be realized by a thread or “lightweight” process. A thread is a stream of instructions that can be scheduled as an independent unit. It is important to understand the difference between a thread and a process. A process contains two kinds of information: resources that are available to the entire process such as program instructions, global data and working directory, and schedulable entities, which include program counters and stacks. A thread is an entity within a process that consists of the schedulable part of the process.

In a single processor system, threads are executed by time-slicing, but shared memory parallel computers can assign threads to different processors.

Pthread Library

There were many thread libraries in the C language for specific shared memory systems. Now, however, the Pthread library is a standard thread library for many systems (Butenhof, 1997). The Pthread API is defined in the ANSI/IEEE POSIX 1003.1-1995 standard, which can be purchased from IEEE.

Listing 3 is an example program to calculate π by using the Pthread library. The program creates a thread using the function `pthread_create()`, then assigns a unique identifier to a variable of type `pthread_t`. The caller provides a function that will be executed by the thread. The function `pthread_exit()` is used to terminate itself. The function `pthread_join()` is analogous to `wait()` for forking, but any thread may join any other thread in the process, that is, there is no parent-child relationship.

As multi-threaded applications execute instructions concurrently, access to process-wide (or interprocess) shared memory requires a mechanism for coordination or synchronization among threads. It is realized by mutual exclusion (mutex) locks. Mutexes furnish the means to guard data structures from concurrent modification. When one thread has locked the mutex, this mechanism precludes other threads from changing the contents of the protected structure until the locker performs the corresponding mutex unlock. Functions `pthread_mutex_init()`, `pthread_mutex_lock()` and `pthread_mutex_unlock()` are used for this purpose.

The compiled executable file is invoked from a command line with two arguments: `n` and the number of threads, which is copied to the global variable `num_threads`. The i th thread of the function `PIworker`, which receives the value i from the original process, calculates a summation for about $n/\text{num_threads}$ times. Each thread adds its result to a global variable `pi`. As the variable `pi` should not be accessed by more than one thread simultaneously, this operation is locked and unlocked by the mutex mechanism.

Listing 3

```
#include <stdio.h>
#include <pthread.h>
int n, num_threads;
```

```
double d, pi;
pthread_mutex_t reduction_mutex;
pthread_t *tid;

void *PIworker(void *arg)
{
    int i, myid;
    double s, x, mypi;
    myid = *(int *)arg;
    s = 0.0;
    for (i=myid+1; i<=n; i+=num_threads) {
        x = (i-0.5)*d;
        s += 4.0/(1.0+x*x);
    }
    mypi = d*s;
    pthread_mutex_lock(&reduction_mutex);
    pi += mypi;
    pthread_mutex_unlock(&reduction_mutex);
    pthread_exit(0);
}

main(int argc, char **argv)
{
    int i;
    int *id;
    n = atoi(argv[1]);
    num_threads = atoi(argv[2]);
    d = 1.0/n;
    pi = 0.0;
    id = (int *) calloc(n,sizeof(int));
    tid = (pthread_t *) calloc(num_threads,
                               sizeof(pthread_t));
    if(pthread_mutex_init(&reduction_mutex,NULL)) {
        fprintf(stderr, "Cannot init lock\n");
        exit(0);
    };
    for (i=0; i<num_threads; i++) {
        id[i] = i;
        if(pthread_create(&tid[i],NULL,
                        PIworker,(void *)&id[i])) {
            exit(1);
        };
    };
    for (i=0; i<num_threads; i++)
        pthread_join(tid[i],NULL);
    printf("pi=%0.15f\n", pi);
}
```

We note that it is not easy to write multi-threaded applications in the C language, even if we use the Pthread library. As the Pthread library was added to the C language later, there are no assurances that original basic libraries are “thread-safe”. The term thread-safe means that a given library function is implemented in such a manner that it can be executed correctly by multiple concurrent threads of execution. We must be careful to use thread-safe functions in multi-thread programming. The Pthread library is mainly used by professional system programmers to support advanced parallel computing technologies such as OpenMP.

Java Threads

The Java language supports threads as one of its essential features (Oaks and Wong, 1999). The Java library provides a `Thread` class that supports a rich collection of methods: for example, the method `start()` causes the thread to execute the method `run()`, the method `join()` waits for the thread to finish execution. The lock-unlock mechanism can be easily realized by the `synchronized` declaration. All fundamental libraries are thread-safe. These features make Java suitable for thread programming.

Listing 4

```
public class PiJavaThread {
    int n, numThreads;
    double pi = 0.0;
    synchronized void addPi(double p) {
        pi += p;
    }
    public PiJavaThread(int nd, int nt) {
        n = nd;
        numThreads = nt;
        Thread threads[] = new Thread[numThreads];
        for (int i=0; i<numThreads; i++) {
            threads[i] = new Thread(new PIworker(i));
            threads[i].start();
        }
        for (int i=0; i<numThreads; i++) {
            try {
                threads[i].join();
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
}
class PIworker implements Runnable {
    int myid;
    public PIworker(int id) {
```

```

        myid = id;
    }
    public void run() {
        double d, s, x;
        d = 1.0/n;
        s = 0.0;
        for (int i=myid+1; i<=n; i+=numThreads) {
            x = (i-0.5)*d;
            s += 4.0/(1.0+x*x);
        }
        addPi(d*s);
    }
}
public static void main(String[] args) {
    PiJavaThread piJavaThread
        = new PiJavaThread(Integer.parseInt(args[0]),
                           Integer.parseInt(args[1]));
    System.out.println(" pi = " + piJavaThread.pi);
}
}

```

Listing 4 is an example program to calculate the value of π using the Java language. This program is almost the same as the Pthread example. As the method declaration for `addPi()` contains the keyword `synchronized`, it can be performed by only one thread; other threads must wait until the `addPi()` method of the currently executing thread finishes.

Although the Java language is designed to be thread-safe and provides several means for thread programming, it is still difficult to write efficient application programs in Java. Java's tools are generally well suited to system programming applications, such as graphical user interfaces and distributed systems, because they provide synchronization operations that are detailed and powerful, but unstructured and complex. They can be considered an assembly language for thread programming. Thus, it is not easy to use them for statistical programming.

OpenMP

8.3.3

OpenMP is a directive-based parallelization technique (Chandra et al., 2001) that supports fork-join parallelism and is mainly for shared memory systems. The MP in OpenMP stands for "Multi Processing". It supports Fortran (77 and 90), C and C++, and is suitable for numerical calculation, including statistical computing. It is standardized for portability by the OpenMP Architecture Review Board (OpenMP Architecture Review Board, 2004). The first Fortran specification 1.0 was released in 1997, and was updated as Fortran specification 1.1 in 1999. New features were added as Fortran specification 2.0 in 2000. Several commercial compilers support OpenMP.

We use the Fortran language for our examples in this section, because Fortran is still mainly used for high-performance computers focused on large numerical computation. Fortran is one of the oldest computer languages and has many reliable and efficient numerical libraries and compilers. The Fortran program for the simple π computation is shown in Listing 5.

We note that C (and C++) are also used for large numerical computations and are now supported to the same extent as Fortran. The following examples can easily be replaced by C programs (except the HPF examples) but we omit them for space reasons.

Listing 5

```
integer n, i
double precision d, s, x, pi
write(*,*) 'n?'
read(*,*) n
d = 1.0/n
s = 0.0
do i=1, n
  x = (i-0.5)*d
  s = s+4.0/(1.0+x*x)
enddo
pi = d*s
write(*,100) pi
100 format(' pi = ', f20.15)
end
```

We can parallelize this program simply by using OpenMP directives (Listing 6).

Listing 6

```
integer n, i
double precision d, s, x, pi
write(*,*) 'n?'
read(*,*) n
d = 1.0/n
s = 0.0
!$OMP PARALLEL PRIVATE(x), SHARED(d)
!$OMP& REDUCTION(+: s)
!$OMP DO
  do i = 1, n
    x = (i-0.5)*d
    s = s+4.0/(1.0+x*x)
  end do
!$OMP END DO
```

```

!$OMP END PARALLEL
    pi = d*s
    write(*,100) pi
100    format(' pi = ', f20.15)
    end

```

Lines started by !\$OMP are OpenMP directives to specify parallel computing. Each OpenMP directive starts with !\$OMP, followed by a directive and, optionally, clauses. For example, “!\$OMP PARALLEL” and “!\$OMP END PARALLEL” encloses a parallel region and all code lexically enclosed is executed by all threads. The number of threads is usually specified by an environmental variable OMP_NUM_THREADS in the shell environment. We also require a process distribution directive “!\$OMP DO” and “!\$OMP END DO” to enclose a loop that is to be executed in parallel. Within a parallel region, data can either be private to each executing thread, or be shared among threads. By default, all data in static extents are shared (an exception is the loop variable of the parallel loop, which is always private). In the example, shared scope is not desirable for *x* and *s*, so we use a suitable clause to make them private: “!\$OMP PARALLEL PRIVATE (*x*, *s*)”. By default, data in dynamic extent (subroutine calls) are private (an exception is data with the SAVE attribute), and data in COMMON blocks are shared.

An OpenMP compiler will automatically translate this program into a Pthread program that can be executed by several processors on shared memory systems.

PVM

8.3.4

PVM (Parallel Virtual Machine) is one of the first widely used message passing programming systems. It was designed to link separate host machines to form a virtual machine, which is a single manageable computing resource (Geist et al., 1994). It is (mainly) suitable for heterogeneous distributed memory systems. The first version of PVM was written in 1989 at Oak Ridge National Laboratory, but was not released publicly. Version 2 was written at the University of Tennessee Knoxville and released in 1991. Version 3 was redesigned and released in 1993. Version 3.4 was released in 1997. The newest minor version, 3.3.4, was released in 2001 (PVM Project Members, 2004).

PVM is freely available and portable (available on Windows and several UNIX systems). It is mainly used in Fortran, C and C++, and extended to be used in many other languages, such as Tcl/Tk, Perl and Python.

The PVM system is composed of two parts: a PVM daemon program (pvmd) and libraries of PVM interface routines. Pvmd provides communication and process control between computers. One pvmd runs on each host of a virtual machine. It serves as a message router and controller, and provides a point of contact, authentication, process control and fault detection. The first pvmd (which must be started by the user) is designated the master, while the others (started by the master) are called slaves or workers.

PVM libraries such as `libpvm3.a` and `libfpvm3.a` allow a task to interface with the `pvm` and other tasks. They contain functions for packing and unpacking messages, and functions to perform PVM calls by using the message functions to send service requests to the `pvm`.

Example Fortran programs are in Listings 7a and 7b.

Listing 7a

```

program pimaster
include '/usr/share/pvm3/include/fpvm3.h'
integer n, i
double precision d, s, pi
integer mytid,numprocs,tids(0:32),status
integer numt,msgtype,info
character*8 arch
write(*,*) 'n, numprocs?'
read(*,*) n, numprocs
call PVMFMYTID(mytid)
arch = '*'
call PVMFSPAWN('piworker',PVMDEFAULT,arch,
$           numprocs,tids,numt)
if( numt .lt. numprocs) then
write(*,*) 'trouble spawning'
call PVMFEXIT(info)
stop
endif
d = 1.0/n
msgtype = 0
do 10 i=0, numprocs-1
call PVMFINITSEND(PVMDEFAULT,info)
call PVMFPACK(INTEGER4, numprocs, 1, 1, info)
call PVMFPACK(INTEGER4, i, 1, 1, info)
call PVMFPACK(INTEGER4, n, 1, 1, info)
call PVMFPACK(REAL8, d, 1, 1, info)
call PVMFSEND(tids(i),msgtype,info)
10 continue
s=0.0
msgtype = 5
do 20 i=0, numprocs-1
call PVMFRECV(-1,msgtype,info)
call PVMFUNPACK(REAL8,x,1,1,info)
s = s+x
20 continue
pi = d*s
write(*,100) pi

```

```

100 format(' pi = ', f20.15)
    call PVMFEXIT(info)
end

```

Listing 7b

```

program piworker
include '/usr/share/pvm3/include/fpvm3.h'
integer n, i
double precision s, x, d
integer mytid,myid,numprocs,msgtype,master,info
call PVMFMYTID(mytid)
msgtype = 0
call PVMFRECV(-1,msgtype,info)
call PVMFUNPACK(INTEGER4, numprocs, 1, 1, info)
call PVMFUNPACK(INTEGER4, myid, 1, 1, info)
call PVMFUNPACK(INTEGER4, n, 1, 1, info)
call PVMFUNPACK(REAL8, d, 1, 1, info)
s = 0.0
do 10 i = myid+1, n, numprocs
    x = (i-0.5)*d
    s = s+4.0/(1.0+x*x)
10 continue
call PVMFINITSEND(PVMDEFAULT,info)
call PVMFPACK(REAL8, s,1,1, info)
call PVMFPARENT(master)
msgtype = 5
call PVMFSEND(master,msgtype,info)
call PVMFEXIT(info)
end

```

Listing 7a is the master program, and Listing 7b is the slave program, and its compiled executable file name should be `piworker`. Both programs include the Fortran PVM header file `fpvm3.h`.

The first PVM call `PVMFMYTID()` in the master program informs the `pvm`d of its existence and assigns a task id to the calling task.

After the program is enrolled in the virtual machine, the master program spawns slave processes by the routine `PVMFSPAWN()`. The first argument is a string containing the name of the executable file that is to be used. The fourth argument specifies the number of copies of the task to be spawned and the fifth argument is an integer array that is to contain the task ids of all tasks successfully spawned. The routine returns the number of tasks that were successfully created via the last argument.

To send a message from one task to another, a send buffer is created to hold the data. The routine `PVMFINITSEND()` creates and clears the buffer and returns

a buffer identifier. The buffer must be packed with data to be sent by the routine `PVMFPACK()`. The first argument specifies the type of data to be packed. The second argument is the first item to be packed, the third is the total number of items to be packed and the fourth is the stride to use when packing. A single message can contain any number of different data types; however, we should ensure that the received message is unpacked in the same way it was originally packed by the routine `PVMFUNPACK()`. The routine `PVMFSEND()` attaches an integer label of `msgtype` and sends the contents of the send buffer to the task specified by the first argument.

After the required data have been distributed to each worker process, the master program must receive a partial sum from each of the worker processes by the `PVMFRECV()` routine. This receives a message from the task specified by the first argument with the label of the second argument and places it into the receive buffer. Note that a value of `-1` for an argument will match with any task id and/or label. The master program expects a label value of `5` on messages from the worker tasks.

The unpacking routine `PVMFUNPACK()` has the same arguments as `PVMFPACK()`. The second argument shows where the first item unpacked is to be stored.

After the sum has been computed and printed, the master task informs the PVM daemon that it is withdrawing from the virtual machine. This is done by calling the routine `PVMFEXIT()`.

The worker program uses the same PVM routines as the master program. It also uses `PVMFPARENT()` routine to find the task id of the master task that spawned the current task.

When we compile Fortran PVM codes, we must link in both the PVM Fortran library and the standard PVM library compiled for the target machine architecture. Before executing the program, the executables of the worker program should be available in a specific directory on all the slave nodes. The default authentication is performed by `rsh` call.

8.3.5 **MPI**

MPI (Message Passing Interface) is the most widely used parallel computing technique. It specifies a library for adding message passing mechanisms to existing languages such as Fortran or C. MPI is mainly used for homogeneous distributed memory systems.

MPI appeared after PVM. PVM was a research effort and did not address the full spectrum of issues: it lacked vendor support, and was not implemented at the most efficient level for a particular hardware. The MPI Forum (Message Passing Interface (MPI) Forum, 2004) was organized in 1992 with broad participation by vendors (such as IBM, Intel, SGI), portability library writers (including PVM), and users such as application scientists and library writers. MPI-1.1 was released in 1995, MPI-1.2 was released in 1997, and MPI-2 was released in 1997.

MPI-1 has several functions that were not implemented in PVM. Communicators encapsulate communication spaces for library safety. Data types reduce copying

costs and permit heterogeneity. Multiple communication modes allow precise buffer management. MPI-1 has extensive collective operations for scalable global communication, and supports process topologies that permit efficient process placement and user views of process layout (Gropp et al., 1999a).

In MPI-2, other functions were added: extensions to the message passing model, dynamic process management, one-sided operations (remote memory access), parallel I/O, thread support, C++ and Fortran 90 bindings, and extended collective operations (Gropp et al., 1999b).

MPI implementations are released from both vendors and research groups. MPICH (MPICH Team, 2004) and LAM/MPI (LAM Team, 2004) are widely used free implementations.

Although MPI has more than 150 routines, many parallel programs can be written using just six routines, only two of which are non-trivial: `MPI_INIT()`, `MPI_FINALIZE()`, `MPI_COMM_SIZE()`, `MPI_COMM_RANK()`, `MPI_SEND()` and `MPI_RECV()`. An example program is shown in Listing 8.

Listing 8

```

include 'mpif.h'
integer n, i
double precision d, s, x, pi, temp
integer myid, numprocs, ierr, status(3)
integer sumtag, sizetag, master
call MPI_INIT(ierr)
call MPI_COMM_SIZE(MPI_COMM_WORLD,numprocs,ierr)
call MPI_COMM_RANK(MPI_COMM_WORLD,myid,ierr)
sizetag = 10
sumtag = 17
master = 0
if (myid .eq. master) then
  write(*,*) 'n?'
  read(*,*) n
  do i = 1, numprocs-1
    call MPI_SEND(n,1,MPI_INTEGER,i,sizetag,
$      MPI_COMM_WORLD,ierr)
  enddo
else
  call MPI_RECV(n,1,MPI_INTEGER,master,sizetag,
$      MPI_COMM_WORLD,status,ierr)
endif
d = 1.0/n
s = 0.0
do i = myid+1, n, numprocs
  x = (i-0.5)*d
  s = s+4.0/(1.0+x*x)

```

```

        enddo
        pi = d*s
        if (myid .ne. master) then
            call MPI_SEND(pi,1,MPI_DOUBLE_PRECISION,
$           master,sumtag,MPI_COMM_WORLD,ierr)
        else
            do i = 1, numprocs-1
                call MPI_RECV(temp,1,MPI_DOUBLE_PRECISION,
$           i,sumtag,MPI_COMM_WORLD,status,ierr)
                pi = pi+temp
            enddo
        endif
        if (myid .eq. master) then
            write(*, 100) pi
100    format(' pi = ', f20.15)
        endif
        call MPI_FINALIZE(ierr)
    end

```

MPI follows the single program-multiple data (SPMD) parallel execution model. SPMD is a restricted version of MIMD in which all processors run the same programs, but unlike SIMD, each processor may take a different flow path in the common program.

If the example program is stored in file `prog8.f`, typical command lines for executing it are

```

f77 -o prog8 prog8.f -lmpi
mpirun -np 5 prog8

```

where the command `mpirun` starts five copies of process `prog8` simultaneously. All processes communicate via MPI routines.

The first MPI call must be `MPI_INIT()`, which initializes the message passing routines. In MPI, we can divide our tasks into groups, called communicators. `MPI_COMM_SIZE()` is used to find the number of tasks in a specified MPI communicator. In the example, we use the communicator `MPI_COMM_WORLD`, which includes all MPI processes. `MPI_COMM_RANK()` finds the rank (the name or identifier) of the tasks running the code. Each task in a communicator is assigned an identifying number from 0 to `numprocs-1`.

`MPI_SEND()` allows the passing of any kind of variable, even a large array, to any group of tasks. The first argument is the variable we want to send, the second argument is the number of elements passed. The third argument is the kind of variable, the fourth is the id number of the task to which we send the message, and the fifth is a message tag by which the receiver verifies that it receives the message it expects. Once a message is sent, we must receive it on another task. The arguments of the routine `MPI_RECV()` are similar to those of `MPI_SEND()`. When we finish

with the message passing routines, we must close out the MPI routines by the call `MPI_FINALIZE()`.

In parallel computing, collective operations often appears. MPI supports useful routines for them. `MPI_BCAST` distributes data from one process to all others in a communicator. `MPI_REDUCE` combines data from all processes in a communicator and returns it to one process. In many numerical algorithms, `SEND/RECEIVE` can be replaced by `BCAST/REDUCE`, improving both simplicity and efficiency. Listing 8 can be replaced by Listing 9 (some parts of Listing 8 are omitted).

Listing 9

```

...
master = 0
if (myid .eq. master) then
  write(*,*) 'n?'
  read(*,*) n
endif
call MPI_BCAST(n,1,MPI_INTEGER,master,
$ MPI_COMM_WORLD,ierr)
d = 1.0/n
s = 0.0
...
enddo
pi = d*s
call MPI_REDUCE(pi,temp,1,MPI_DOUBLE_PRECISION,
$ MPI_SUM,master,MPI_COMM_WORLD,ierr)
pi = temp
if (myid .eq. master) then
  write(*, 100) pi
...

```

In distributed shared memory systems, both OpenMP and MPI can be used together to use all the processors efficiently. Again, Listing 8 can be replaced by Listing 10 (the same parts of Listing 8 are omitted) to use OpenMP.

Listing 10

```

...
d = 1.0/n
s = 0.0
!$OMP PARALLEL PRIVATE(x), SHARED(d)
!$OMP& REDUCTION(+: s)
!$OMP DO
  do i = myid+1, n, numprocs

```

```

        x = (i-0.5)*d
        s = s+4.0/(1.0+x*x)
    enddo
!$OMP END DO
!$OMP END PARALLEL
    pi = d*s
    if (myid .ne. master) then
        ...

```

8.3.6 HPF

HPF (High Performance Fortran) is a Fortran 90 with further data parallel programming features (Koelbel et al., 1993). In data parallel programming, we specify which processor owns what data, and the owner of the data does the computation on the data (Owner-computes rule).

Fortran 90 provides many features that are well suited to data parallel programming, such as array processing syntax, new functions for array calculations, modular programming constructs and object-oriented programming features.

HPF adds additional features to enable data parallel programming. We use compiler directives to distribute data on the processors, to align arrays and to declare that a loop can be calculated in parallel without affecting the numerical results. HPF also has a loop control structure that is more flexible than DO, and new intrinsic functions for array calculations.

The High Performance Fortran Forum (HPFF) (High Performance Fortran Forum, 2004) is a coalition of industry, academic and laboratory representatives, and defined HPF 1.0 in 1993. HPF 1.1 was released in 1994 and HPF 2.0 was released in 1997. Several commercial and free HPF compilers are now available.

Listing 11 is an example program for calculating π in HPF.

Listing 11

```

integer n, i
double precision d, s, pi
double precision, dimension (:),
    $          allocatable :: x, y
!HPF$ PROCESSORS procs(4)
!HPF$ DISTRIBUTE x(CYCLIC) ONTO procs
!HPF$ ALIGN y(i) WITH x(i)
write(*,*) 'n?'
read(*,*) n
allocate(x(n))
allocate(y(n))

```

```

      d = 1.0/n
!HPF$ INDEPENDENT
      FORALL (i = 1:n)
        x(i) = (i-0.5)*d
        y(i) = 4.0/(1.0 + x(i)*x(i))
      end FORALL
      pi = d*SUM(y)
      write (*, 100) pi
100  format(' pi = ', f20.15)
      deallocate(x)
      deallocate(y)
      end

```

!HPF\$ is used for all HPF compiler directives. We note that this is a comment to non-HPF compilers and is ignored by them. The PROCESSORS directive specifies the shape of the grid of abstract processors. Another example “!HPF\$ PROCESSORS exprocs(6,2)” specifies a 6×2 array of 12 abstract processors labelled exprocs.

The DISTRIBUTE directive partitions an array by specifying a regular distribution pattern for each dimension ONTO the arrangement of abstract processors. The CYCLIC pattern spreads the elements one per processor, wrapping around when it runs out of processors, i.e., this pattern distributes the data in the same way that the program in Listing 8 performs. Another pattern is BLOCK, which breaks the array into equal-sized blocks, one per processor. The rank of the abstract processor grid must be equal to the number of distributed axes of the array.

The ALIGN directive is used to specify relationships between data objects. In the example program, elements of x and y that have the same index are placed on the same processor.

The INDEPENDENT directive informs the compiler that in the execution of the FORALL construct or the do loop, no iteration affects any other iteration in any way.

The FORALL statement is a data parallel construct that defines the assignment of multiple elements in an array but does not restrict the order of assignment to individual elements. Note that the do loop executes on each element in a rigidly defined order.

The SUM intrinsic function performs reduction on whole arrays.

We may compare HPF with OpenMP, because both systems use compiler directives in a standard language (Fortran) syntax. In OpenMP, the user specifies the distribution of iterations, while in HPF, the user specifies the distribution of data. In other words, OpenMP adopts the instruction parallel programming model while HPF adopts data parallel programming model. OpenMP is suitable for shared memory systems whereas HPF is suitable for distributed memory systems.

8.4 Parallel Computing in Statistics

8.4.1 Parallel Applications in Statistical Computing

The most important thing in parallel computing is to divide a job into small tasks for parallel execution. We call the amount of independent parallel processing that can occur before requiring some sort of communication or synchronization the “granularity”. Fine granularity may allow only a few arithmetic operations between processing one message and the next, whereas coarse granularity may allow millions. Although the parallel computing techniques described above can support programming of any granularity, coarse granularity is preferable for many statistical tasks. Fine granularity requires much information exchange among processors and it is difficult to write the required programs. Fortunately, many statistical tasks are easily divided into coarse granular tasks. Some of them are embarrassingly parallel.

In data analysis, we often wish to perform the same statistical calculations on many data sets. Each calculation for a data set is performed independently from other data sets, so the calculations can be performed simultaneously. For example, Hegland et al. (1999) implemented the backfitting algorithm to estimate a generalized additive model for a large data set by dividing it into small data sets, fitting a function in parallel and merging them together. Beddo (2002) performed parallel multiple correspondence analysis by dividing an original data set and merging their calculation results.

Another embarrassingly parallel example is a simulation or a resampling computation, which generates new data sets by using a random number generating mechanism based on a given data set or parameters. We calculate some statistics for those data sets, repeat such operations many times and summarize their results to show empirical distribution characteristics of the statistics. In this case, all calculations are performed simultaneously except the last part. Beddo (2002) provided an example of bootstrapping from parallel multiple correspondence analysis.

We must be careful that random numbers are appropriately generated in parallel execution. For example, random seeds for each process should all be different values, at least. SPRNG (Mascagni, 1999) is a useful random number generator for parallel programming. It allows for the dynamic creation of independent random number streams on parallel machines without interprocessor communication. It is available in the MPI environment and the macro `SIMPLE_SPRNG` should be defined to invoke the simple interface. Then the macro `USE_MPI` is defined to instruct SPRNG to make MPI calls during initialization. Fortran users should include the header file `sprng_f.h` and call `sprng()` to obtain a double precision random number in $(0, 1)$. In compiling, the libraries `liblcg.a` and the MPI library should be linked.

The maximum likelihood method requires much computation and can be parallelized. Jones et al. (1999) describes a parallel implementation of the maximum likelihood estimation using the EM algorithm for positron emission tomography

image reconstruction. Swann (2002) showed maximum likelihood estimation for a simple econometric problem with Fortran code and a full explanation of MPI. Malard (2002) solved a restricted maximum likelihood estimation of variance-covariance matrix by using freely available toolkits: the portable extensible toolkit for scientific computation (PETSc) and the toolkit for advanced optimization (TAO) (Balay et al., 2001) which are built on MPI.

Optimization with dynamic programming requires much computation and is suitable for parallel computing. Hardwick et al. (1999) used this technique to solve sequential allocation problems involving three Bernoulli populations. Christofides et al. (1999) applied it to the problem of discretizing multidimensional probability functions.

Racine (2002) demonstrated that kernel density estimation is also calculated efficiently in parallel.

Parallel Software for Statistics

8.4.2

Several commercial and non-commercial parallel linear algebra packages that are useful for statistical computation are available for Fortran and/or C. We mention two non-commercial packages with freely available source codes: ScaLAPACK (Blackford et al., 1997) supports MPI and PVM, and PLAPACK (van de Geijin, 1997) supports MPI. Murphy et al. (1999) described the work to transfer sequential libraries (Gram-Schmidt orthogonalization and linear least squares with equally constraints) to parallel systems by using Fortran with MPI.

Although we have many statistical software products, few of them have parallel features. Parallel statistical systems are still at the research stage. Bull et al. (1999) ported a multilevel modeling package MLn into a shared memory system by using C++ with threads. Yamamoto and Nakano (2002) explained a system for time series analysis that has functions to use several computers via Tkpvml, an implementation of PVM in the Tcl/Tk language.

The statistical systems R (The R Development Core Team, 2004) and S (Chambers, 1998) have some projects to add parallel computing features. Temple Lang (1997) added thread functions to S. PVM and MPI are directly available from R via the rpvm (Li and Rossini, 2001) and Rmpi (Yu, 2002) packages. They are used to realize the package “snow” (Rossini et al., 2003), which implements simple commands for using a workstation cluster for embarrassingly parallel computations in R. A simple example session is:

```
> cl <- makeCluster(2, type = "PVM")
> clusterSetupSPRNG(cl)
> clusterCall(cl, runif, 3)
[[1]]
[1] 0.749391854 0.007316102 0.152742874

[[2]]
[1] 0.8424790 0.8896625 0.2256776
```

where a PVM cluster of two computers is started by the first command and the SPRNG library is prepared by the second command. Three uniform random numbers are generated on each computer and the results are printed by the third command.

The statistical system “Jasp” (Nakano et al., 2000) is implementing experimental parallel computing functions via network functions of the Java language (see also <http://jasp.ism.ac.jp/>).

References

- Amdahl, G. M. (1967). Validity of the single-processor approach to achieving large scale computing capabilities. In *AFIPS Conference Proceedings*, volume 30, pages 483–485.
- Balay, S., Buschelman, K., Gropp, W. D., Kaushik, D., Knepley, M., McInnes, L. C., Smith, B. F., and Zhang, H. (2001). PETSc home page. <http://www.mcs.anl.gov/petsc>.
- Beddo, V. (2002). *Applications of parallel programming in Statistics*. Ph.D. dissertation, University of California, Los Angeles. http://theses.stat.ucla.edu/19/parallel_programming_beddo.pdf.
- Blackford, L., Choi, J., Cleary, A., D’Azevedo, E., Demmel, J., Dhillon, I., Dongarra, J., Hammarling, S., Henry, G., Petitet, A., Stanley, K., Walker, D., and Whaley, R. C. (1997). *ScaLAPACK Users’ Guide*. SIAM Press.
- Bull, J. M., Riley, G. D., Rasbash, J., and Goldstein, H. (1999). Parallel implementation of a multilevel modelling package. *Computational Statistics & Data Analysis*, 31(4):457–474.
- Butenhof, D. R. (1997). *Programming with POSIX Threads*. Addison Wesley.
- Chambers, J. M. (1998). *Programming with Data: A Guide to the S Language*. Springer-Verlag.
- Chandra, R., Dagum, L., Kohr, D., Maydan, D., McDonald, J., and Menon, R. (2001). *Parallel Programming in OpenMP*. Morgan Kaufman.
- Christofides, A., Tanyi, B., Christofides, D., Whobrey, D., and Christofides, N. (1999). The optimal discretization of probability density functions. *Computational Statistics & Data Analysis*, 31(4):475–486.
- Flynn, M. (1966). Very high-speed computing systems. *Proc. IEEE*, 54(12):1901–1909.
- Geist, A., Beguelin, A., Dongarra, J., Jiang, W., Manchek, R., and Sunderam, V. S. (1994). *PVM: Parallel Virtual Machine: A Users’ Guide and Tutorial for Networked Parallel Computing*. MIT Press.
- Gropp, W., Lusk, E., and Skjellum, A. (1999a). *Using MPI: Portable Parallel Programming with the Message-Passing Interface, 2nd Edition*. MIT Press.
- Gropp, W., Lusk, E., and Thakur, R. (1999b). *Using MPI-2: Advanced Features of the Message-Passing Interface*. MIT Press.
- Gustafson, J. L. (1988). Reevaluating amdahl’s law. *Comm. ACM*, 31(5):532–533.

- Hardwick, J., Oehmke, R., and Stout, Q. F. (1999). A program for sequential allocation of three bermoulli populations. *Computational Statistics & Data Analysis*, 31(4):397–416.
- Hegland, M., McIntosh, I., and Turlach, B. A. (1999). A parallel solver for generalized additive models. *Computational Statistics & Data Analysis*, 31(4):377–396.
- High Performance Fortran Forum (2004). HPF: The high performance fortran home page. <http://www.crpc.rice.edu/HPFF/>.
- Jones, H., Mitra, G., Parkinson, D., and Spinks, T. (1999). A parallel implementation of the maximum likelihood method in positron emission tomography image reconstruction. *Computational Statistics & Data Analysis*, 31(4):417–439.
- Koelbel, C. H., Loveman, D. B., Schreiber, R. S., Steele, J. G. L., and Zosel, M. E. (1993). *The High Performance Fortran Handbook*. MIT Press.
- LAM Team (2004). LAM/MPI parallel computing. <http://www.lam-mpi.org/>.
- Li, N. and Rossini, A. (2001). RPVM: Cluster statistical computing in R. *R News*, 1(3):4–7. <http://CRAN.R-project.org/doc/Rnews/>.
- Malard, J. M. (2002). Parallel restricted maximum likelihood estimation for linear models with a dense exogenous matrix. *Parallel Computing*, 28(2):343–353.
- Mascagni, M. (1999). SPRNG: A scalable library for pseudorandom number generation. In Spanier, J. et al., editor, *Proceedings of the Third International Conference on Monte Carlo and Quasi Monte Carlo Methods in Scientific Computing*. Springer-Verlag.
- Message Passing Interface (MPI) Forum (2004). Message passing interface (MPI) forum home page. <http://www.mpi-forum.org/>.
- MPICH Team (2004). MPICH – A portable mpi implementation. <http://www-unix.mcs.anl.gov/mpi/mpich/>.
- Murphy, K., Clint, M., and Perrott, R. H. (1999). Re-engineering statistical software for efficient parallel execution. *Computational Statistics & Data Analysis*, 31(4):441–456.
- Nakano, J., Fujiwara, T., Yamamoto, Y., and Kobayashi, I. (2000). A statistical package based on Pnuts. In Bethlehem, J. G. and van der Heijden, P. G. M., editors, *COMPSTAT 2000 Proceedings in Computational Statistics*, pages 361–366. Physica-Verlag.
- Oaks, S. and Wong, H. (1999). *Java Threads, 2nd edition*. O’Reilly.
- OpenMP Architecture Review Board (2004). OpenMP: Simple, portable, scalable SMP programming. <http://www.openmp.org/>.
- PVM Project Members (2004). PVM: Parallel virtual machine. http://www.csm.ornl.gov/pvm/pvm_home.html.
- Racine, J. (2002). Parallel distributed kernel estimation. *Computational Statistics & Data Analysis*, 40(2):293–302.
- Rossini, A., Tierney, L., and Li, N. (2003). Simple parallel statistical computing in R. UW Biostatistics working paper series, Paper 193, University of Washington. <http://www.bepress.com/uwbiostat/paper193>.
- Schervish, M. J. (1988). Applications of parallel computation to statistical inference. *J. Amer. Statist. Assoc.*, 83:976–983.

-
- Sterling, T., Salmon, J., Becker, D. J., and Savarese, D. F. (1999). *How to Build a Beowulf: A Guide to the Implementation and Application of PC Clusters*. MIT Press.
- Swann, C. A. (2002). Maximum likelihood estimation using parallel computing: An introduction to MPI. *Computational Economics*, 19:145–178.
- Tanenbaum, A. S. (2001). *Modern Operating Systems, 2nd Edition*. Prentice Hall.
- Temple Lang, D. (1997). *A multi-threaded extension to a high level interactive statistical computing environment*. Ph.D. dissertation, University of California, Berkeley. <http://cm.bell-labs.com/stat/doc/multi-threaded-S.ps>.
- The R Development Core Team (2004). The R project for statistical computing. <http://www.r-project.org/>.
- Tuomi, I. (2002). The lives and death of moore's law. *First Monday*, 7(11). http://firstmonday.org/issues/issue7_11/tuomi/index.html.
- van de Geijin, R. A. (1997). *Using PLAPACK*. MIT Press.
- Yamamoto, Y. and Nakano, J. (2002). Distributed processing functions of a time series analysis system. *Journal of the Japanese Society of Computational Statistics*, 15(1):65–77.
- Yu, H. (2002). Rmpi: Parallel statistical computing in R. *R News*, 2(2):10–14. <http://CRAN.R-project.org/doc/Rnews/>.

Statistical Databases

II.9

Claus Boyens, Oliver Günther, Hans-J. Lenz

9.1	<i>Introduction</i>	269
9.2	<i>Fundamentals of Data Management</i>	270
	File Systems	270
	Relational Database Systems (RDBS)	271
	Data Warehouse Systems (DWS)	273
9.3	<i>Architectures, Concepts and Operators</i>	275
	Architecture of a Database System for OLTP	275
	Architecture of a Data Warehouse	276
	Concepts (ROLAP, MOLAP, HOLAP, Cube Operators)	276
	Summarizability and Normal Forms	279
	Comparison of Terminologies	280
9.4	<i>Access Methods</i>	281
	Views (Virtual Tables)	281
	Tree-based Indexing	281
	Bitmap Index Structures	282
9.5	<i>Extraction, Transformation and Loading (ETL)</i>	284
9.6	<i>Metadata and XML</i>	285
9.7	<i>Privacy and Security</i>	287
	Preventing Disclosure of Confidential Information	287
	Query Set Restriction	287
	Data Perturbation	289
	Disclosure Risk versus Data Utility	289

Introduction

Most data collected in statistics and science is still stored in simple *flat files*, usually data matrices with rows identified by a case identifier (`case_id`), columns corresponding to attributes (variables), and numerical data types for the elements of each matrix due to coding of all attributes involved. Each row (tuple) carries the (coded) values of the attributes, besides the `case_id`. Due to a suitable encoding that maps a natural domain to numerical ones, all matrix entries have a numeric data type. The scales of the attributes may of course be quite different.

A simple example is given by census data stored at statistical offices in files according to a schema like

```
census_questionnaire (case_id, age-group, gender,
  profession, ...).
```

While science gains their data from experiments, statistical agencies collect their data still mostly off-line from surveys, reports or census. Industry and services get their data on-line from their business process management, i.e., from their logistical, production and administrative transactions. A typical example is sales data, which may be represented by a schema like

```
sales (transaction_id, customer_id, date,
  product_name, quantity, price, amount).
```

Such data is called *microdata*, since it is kept in its original form and is not divisible but atomic. In the business area such data is labeled as *on-line transaction data* because it is subject to frequent updates and is the basis for continuous business transactions. The use of a simple file system to store microdata is rarely a good choice because of a lack of safety and integrity, and retrieval problems. Such data should rather be stored as tables of a relational database. A *database management system* (DBMS) asserts safety, integrity and retrieval flexibility. For instance, a query like “Find prices and amount of all sales since year 2001 where customer 007 with product 4711 is involved” can be simply stated in *structured query language* (SQL) as

```
SELECT price, amount FROM sales
WHERE year >= 2001
AND customer_id = 007
AND product_name = 4711;.
```

It is interesting to note that SQL provides for a set of query operators that is relationally complete. One may thus process any reasonable query as long as it does not involve “transitive closure”, i.e. a potentially infinite loop based on some logical inference (such as a part-of hierarchy).

Macrodata is derived from microdata by applying statistical functions, aggregation and grouping, and consequently has a larger granularity. For example, a business analyst might be interested in a *three-way table* (*data cube*) of total

sales classified by month and year, `customer_id` and `product_name`. Such a retrieval can be achieved on sales by the command:

```
SELECT SUM(sales), date.month, date.year, customer_id,
       product_name
FROM sales Group BY date.month, date.year,
       customer_id, product_name;.
```

This type of activities is coined *on-line analytical operations* (OLAP), which expresses clearly its objective, i.e. a statistical analysis of data for planning, decision support, and controlling.

As we shall see later there does not exist a clear boundary between retrieval and statistical modeling. However, a statistical function like sum (or average) must be selected for a specific query, which does imply modeling. Consequently, there will not exist a closed set of operators on such multi-way tables. Moreover, there are two further problems involved. First of all, which data structure of such kind of data is efficient, and secondly, what kind of background information is needed, to assist the management and the interpretation of real data? This leads to discuss *metadata* as data about real data and functions. Modern database management systems encapsulate metadata in a *repository* (integrated metadata database).

In the following we are first concerned with some fundamentals of data management. Then we turn to the architecture of a statistical database or a data warehouse (DW) and some concepts related to it. We pay special attention to conceptual data structures and related operators involved, the summarizability problem, and hierarchical attributes. We discuss metadata, access methods and “extraction, transformation and loading” (ETL). We close with metadata and extensible markup language (XML), and privacy.

Fundamentals of Data Management

9.2

We start our discussion with file systems, have a look at database systems (DBSs) useful to store transaction or microdata, and finally turn to DWs which host macrodata either in a real (materialized) or virtual form.

File Systems

9.2.1

Data is classically stored in *files*. Files can be viewed as a conceptually related set R of records, which are represented by a given record type, see Wirth (1986), and an access mode (direct or sequential). If the records have a numeric type for each of its fields and the mode is sequential, then a data matrix can be stored in a sequential file. A collection of such files is called a file system (FS), if there exist logical relations between the files $f \in FS$, a set of constraints on FS and application software. Typical applications in statistics are simple surveys like price surveys, where in most cases only one file is needed. A more complex file system

is compulsory if, for instance, stratified or panel sampling designs are considered, where various sampling periods, areas, objects and units (carriers of interest) are involved. Moreover, relational data mining, as described by Dzeroski and Lavrac (2001) and Wrobel (2001), is devoted to such data structures.

File systems are appropriate if only single user-access and weakly logically connected files with simple constraints are effective. Note that application programs must be specially tailored to execute queries, and to achieve data safety and security. This implies data dependence between the software and the files referenced, which reduces the program's flexibility with respect to structural changes of the data structure. These pitfalls can be overcome by DBSs.

9.2.2 Relational Database Systems (RDBS)

Multi-user access, complex data structures and logical restrictions ask for a *relational database system* (RDBS). It consists of a set T of relations (flat tables) together with a set S of corresponding schemas and a set C of constraints, a database management system and application software. A database schema describes the attributes (variables) of a specific table, its data types and roles. To avoid redundancy and anomalies during insert, delete or update transactions, those tables should be transformed into a “normal form”, see Elmasri and Navathe (1999). As an example, we take a Census. When we look at the RDBS ‘Census’ from a conceptual point of view, there are four table schemas involved: Census-questionnaire, household, dwelling, and employment. We shall consider only the first two in some detail, and select only some few attributes for the sake of brevity. The first schema is

```
census_questionnaire(case_id, age-group, gender,
                    profession, ...).
```

Its first three attributes are numeric and the fourth one is of type ‘string’. The attribute `case_id` acts as a primary key, i.e., the remaining attributes are functionally dependent on it. Because a key attribute uniquely identifies any tuple (record) of the corresponding table (set of tuples), there is one constraint among others saying that duplicates in a given table are not allowed. In order to mention just one further constraint, the domain of the identifier `case_id` may be restricted to the set of positive integers.

The next schema is

```
household (household_id, case-id, role,...).
```

The first two attributes have a numeric domain, while `role` is of type ‘string’ with the value set {“member”, “owner”}. Of course, we have again the constraint that duplicates are not allowed, but we need at least one further restriction to ensure reference integrity, i.e., whenever there exist entries of people grouped together in a household, each of their corresponding records in `census_questionnaire` has to exist.

Last but not least, we reconsider our sales example from the introduction. The schema is

```
sales (transaction_id, customer_id, date, product_name,
      quantity, price, amount)
```

The primary key is `transaction_id`, which implies that only one product can be part of any transaction. Evidently, this scheme is not normalized, because `price` depends on `product_name` besides of `transaction_id`, and `amount = quantity × price`. The relation itself is of degree (number of attributes) seven. The six attributes `customer_id`, `date`, ..., `amount` span a six-dimensional data space, where each tuple has six data elements, and is identified by its corresponding `transaction_id`. We represent four tuples in Table 9.1 to illustrate the difference between a schema and its corresponding relation (table). We use abbreviations in the header of the table `sales`.

Table 9.1. The relational table `sales` of degree 7 and cardinality 4

Transaction_id	customer_id	Date	Product_name	Quantity	Price	Amount
015	A	4 Jan 97	Tennis Shoes	200	95	19,000.00
018	A	4 Jan 97	Tennis Balls	300	1.50	450.00
004	A	3 Jan 97	Tennis Nets	350	27	9450.00
009	C	3 Jan 97	Tennis Shoes	100	95	9500.00
...

The need of various users for different data can be satisfied by the concept of virtual relations (views), which can be created on top of an existing DBS.

Note that the term “table” used in a relational database to store such information is quite different from the tables statisticians use for the same purpose. Table 9.2 shows the representation of the same information in a different table structure that allows the natural computation of aggregates along rows and columns (“margin sums” etc.). Note that this table structure cannot be mapped directly into a relational database context due to the margins (Total or ALL), see Gray et al. (1996).

Let us close this example with a discussion of the background information needed. We mentioned above metadata like schema names, attribute names, data types, roles (key versus non key) of attributes, constraints etc. All this can be considered as technical metadata. Moreover, we need further metadata of a semantic and statistical type. Take for instance the attributes `quantity`, `price` and `amount`. What is their definition? As far as `amount` is concerned we have “`amount = quantity * price`”. Furthermore, we need the corresponding *measurement units* which may be units, €/unit and €. As far as data collection at Statistical Offices is concerned, we may need information about the *periodicity* of data surveys like ‘annual’, ‘quarterly’ or ‘monthly’. With respect to data analysis we may be interested in the *measurement scale*. While `product_name` has a nominal scale allowing only operations like ‘equal’ and ‘not equal’, the attributes `quantity`, `price` and

Table 9.2. sales data in the form of the three-way statistical table `total_sales`

3 Jan 1997	Tennis shoes	Tennis balls	Tennis nets
Customer A	0	0	350
Customer B	0	0	0
Customer C	100	0	0
Total	100	0	350
4 Jan 1997	Tennis shoes	Tennis balls	Tennis nets
Customer A	300	400	450
Customer B	1100	1100	800
Customer C	600	1600	350
Total	2350	3400	1900

amount have a metric scale allowing for all basic numerical operations. There exist further ambiguities. For example, the *generation mode* of the attribute `sales` may have the categories ‘real’, ‘simulated’ or ‘forecasted’. There may exist further vagueness about `sales` of category ‘real’ unless its *update state* is set to ‘final’, and not to ‘provisional’.

9.2.3 Data Warehouse Systems (DWS)

A *data warehouse system* (DWS) consists of a (replicated) micro database, a set of materialized or virtual multi-way tables (*data cubes*) needed to represent macro (pre-aggregated and grouped) data, a *data warehouse management system* (DWMS), and a repository, which stores all required technical, statistical and semantic metadata.

As an example of a data cube, we remind the reader of the three-way table presented above:

```
total_sales (date.month, date.year, customer_id,
            product_name, sum(sales)).
```

This table is represented in a relational form, where `date`, `customer_id` and `product_name` are concatenated as a primary key. These attributes are called *dimensions*. Evidently, the non-key attribute `sum(sales)` is fully dependent upon this key, i.e. given the values of `date.month`, `date.year`, `customer_id`, `product_name` there exist one and only one value of `sum(sales)` if missing values (null values) are excluded.

Views are useful again and can be provided by joining cubes or sub-cubes in combination with table projections to lower dimensions. It is worthwhile considering separately the attributes `sum(sales)`, `date` and `product_name`. The first attribute is sometimes called summary attribute and is composed of the statistical sum applied to the attribute `sales`, see Shoshani (1997). This operation is feasible

because the function `sum` and the attribute `sales` have an identical data type, i.e., a metric type. Moreover, the attribute `sales` is of attribute type `flow`, but not `stock`. While summarizing over flows (rates) is reasonable, such an operation over stocks like ‘number of customers’ is nonsense. Evidently, such and further integrity constraints must be effective for a DWS, in order to protect the naive user from nonsense queries. This is extremely important for data warehousing, because contrary to database queries, in DWS the application of statistical functions is an inherent part of any query.

Furthermore, there exists a specific problem related to `date`. This attribute can be decomposed into `month` and `year` but these components are functionally dependent, i.e., for a given month of a calendar year the year is fixed. We thus have $(month, year) \rightarrow year$ as a functional dependency. Therefore only one dimension called `date` is used for the two attributes `month` and `year` in the data cube above. There may be further temporal levels like hour, day, month, quarter and year. Such hierarchical attributes are called taxonomies and need special attention, see Lehner et al. (1998). It is quite remarkable that all dimensions can be allocated to three principal groups: time, location and subject. This is called the 3D-principle, see Lenz (1994).

Let us have a further look at *taxonomies* that are unbalanced and asymmetric. This may happen in case of a product or regional hierarchy. In our running example the subgroups `tennis shoes` and `balls` may be grouped together as `product_group1`, while `tennis nets` build-up `product_group2`, but are free of sub-grouping. Both groups 1 and 2 build the root group `product_all`. As subgroups exist only for shoes and balls, subgroups are no longer functionally dependent on `product_name`, but only weakly functionally dependent, see Lehner et al. (1998), Fig. 9.1. This implies that queries, which involve sub-grouping over products, are not feasible and must be refused. Further pitfalls of operations on a data-cube are given in Lenz and Shoshani (1997) and Lenz and Thalheim (2001).

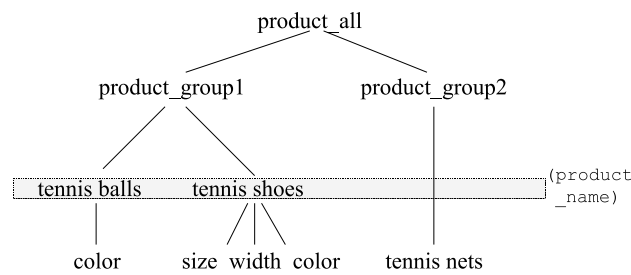


Figure 9.1. The product taxonomy with a weak functional dependency

This discussion shows that real data without metadata is more or less useless especially for on-line analytical processing (OLAP). A repository with metadata has become a prerequisite of any DBS engineering and sound data analysis.

Architectures, Concepts and Operators

We first consider the architecture of micro or operational data used for *online transaction processing* (OLTP), and then illustrate the different architecture of macro or analytical data used for decision support and its relation to operational data, see OLAP. We note that the key features of a DBS for OLTP data are: transaction-oriented, measurement- or record-based, real time processing of inserts, deletes and updates of records. In contrary, a DWS for OLAP data is characterized by the features: subject-oriented, integrated and aggregated, calendar or fiscal period related, and non-volatile, see Inmon (1992).

Architecture of a Database System for OLTP

The architecture of DBS can be represented by the quintuple (data sources, application server, DB server with a DBMS, application server, DB and repository); see also Fig. 9.2. As mentioned above, business processes act as data sources in commercial systems, while at statistical offices data is supplied by surveys, periodic reports or a census. Similarly, in science the data is generated by observations or measurements collected by field or simulation experiments. We represent the architecture in Fig. 9.2.

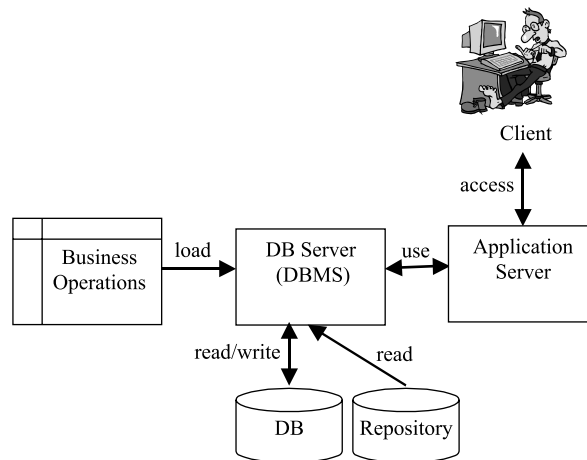


Figure 9.2. Architecture of a DBS used to manage and query operational data

As an example from business we consider a company, which manages wages and salaries of its employees. The data is generated by bookkeeping, the DBMS administers the real and metadata, processes queries, and controls transactions. The application server is responsible for running the software for wage and salary computation, while the client is used as a presentation layer.

Architecture of a Data Warehouse

9.3.2

The main components of the architecture of any OLAP application are heterogeneous data sources S like internal or external databases or files, an OLAP server with DWMS, DW, Repository and Data Marts, and OLAP clients. The DWMS is responsible for load management, query management and warehouse management.

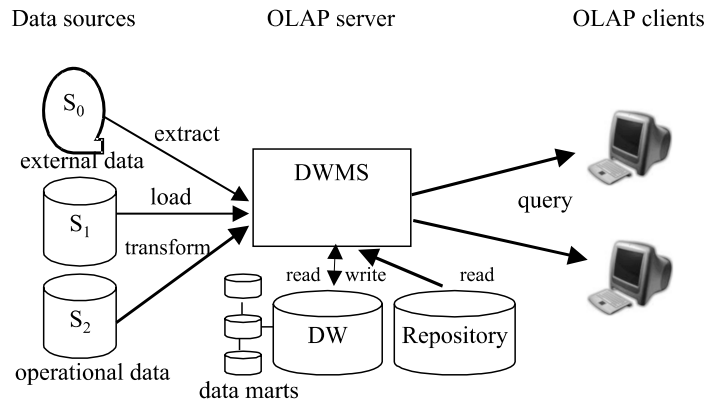


Figure 9.3. DW Architecture

The DW (see Fig. 9.3) incorporates data replications, archived data and aggregated data stored as data cubes. The departmental view on the whole data is given by subsets of the data cube, called data marts.

As can be seen from Fig. 9.3, analytical processing is concerned with data from various data sources, i.e., external or internal (operational) data. These sources are integrated by ETL in data marts in a unified manner. The data marts can be viewed as collections of data cubes.

There exist two types of OLAP clients:

1. stand-alone applications like spreadsheets with a DW interface, and
2. Web clients that use an Internet browser and often applets.

Concepts (ROLAP, MOLAP, HOLAP, Cube Operators)

9.3.3

As we have seen above, the schema of a data cube consists of a cube identifier (name), a list of identifying attributes called dimensions and a statistical function like min, max, count (frequency), sum, avg (arithmetic mean) applied to a summary attribute. Furthermore, the data types of the attributes and integrity constraints must be given. As an example we take from above the data cube "sales cross-classified by (month, year), customer and product":

```
total_sales (date.month, date.year, customer_id,
            product_name, sum(sales)).
```

Evidently, the dimensions span a three-dimensional space on which the statistical function $\text{sum}(\text{sales})$ is defined. The corresponding data types are date (mm.yyyy), integer, string and decimal.

Relational OLAP (ROLAP)

In the following we turn to the conceptual mapping of a data cube into a relational database schema. This approach is called *ROLAP* for Relational OLAP, see Raden (1996). There exist two schemas, star and snowflake schemas. As illustrated in Fig. 9.4, the star schema uses two different types of schemas, which refer to two types of corresponding tables:

1. a *fact table* with a primary key reference to each dimension and the facts which are composed of at least a statistical function and a summary attribute.
2. a *dimension table* for each dimension with a primary key and a level indicator for each entry of a hierarchical attribute.

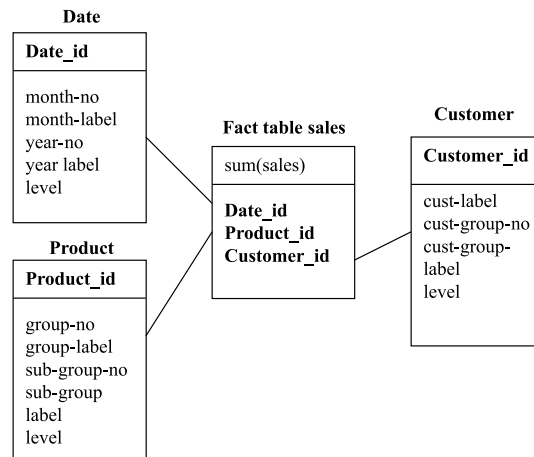


Figure 9.4. Star schema of a three-dimensional data cube (one fact table, three dimension tables; the product hierarchy is assumed to have two levels)

The star schema models all kind of hierarchical attributes including parallel hierarchies, see Lehner et al. (1997). The schema is not normalized as becomes obvious, for example, from the dimension table Date. The attributes month and year are nested, which implies some redundancy. For small or medium-sized data volumes, such schemas have a sufficient performance because join operations are only necessary between the fact table and the related dimension tables.

In order to normalize tables by level attributes, the snowflake schema was introduced. Instead of modelling each dimension by one table, a table is created for each *level* of a hierarchical attribute. The schemas involved are related by identifiers, which play the role either of a primary or a foreign key. In Fig. 9.5 we display only the normalized dimension tables Month and Year and the fact

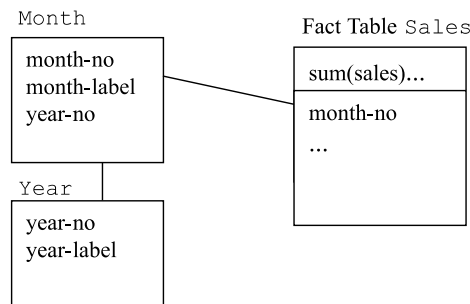


Figure 9.5. Data cube Sales represented (fractionally) as a snowflake schema

table Sales. The identifiers are `month-no` in the fact table and dimension table Month and `year-no` in the dimension table Year.

It can be shown that the normalization is lossless by applying an inner join to the tables of a snowflake schema.

Other Storage Modes (MOLAP, HOLAP)

The above conceptual model of a star or snowflake schema may lead to the wrong conclusion that data cubes are exclusively represented by a relational data model approach. There exist further storage modes, which are in use.

The main advantage of ROLAP lies in the reliability, security and ease of loading of the DW based on *Relational DBMS* (RDBMS) technology. As was mentioned above, this is achieved due to the mapping of facts into a normalized relation and dimension into a mostly non-normalized relation of a relational database. As the set of statistical functions in SQL is too restrictive, some of the functionality of OLAP must be added to the application server. An example is to find the *top-ten* among all products sold in a given period.

Multi-dimensional OLAP (MOLAP) makes use of specially tailored data structures like arrays and associated dimension lists or bitmaps. The operational data is extracted and stored as aggregates in those structures. The performance is acceptable for up to medium-sized data sets (< 1 Gbyte). There exists a multi-dimensional query language called MDX (Multidimensional Expressions), see Microsoft (1998). “XML for Analysis” defines a standardized programming interface for an OLAP server, see <http://www.xmla.org>. An OLAP client encodes a query of a data cube and inserts it into a XML document, which specifies the method “execute” and the accompanying parameters according to the “Simple Object Access Protocol” (SOAP). This document is transmitted over the Internet based on the “Hypertext Transfer Protocol” (HTTP). After decoding the OLAP server executes the query, and sends the data back in a XML document to the client according to SOAP. For further details see Messerschmidt and Schweinsberg (2003). MOLAP has the disadvantage of “miss hits” if a data cube cannot be stored fully in-core and an access to a second storage device is necessary. Moreover, array compression or sparse array handling is needed because mostly the data cube or, equivalently, the arrays are sparse.

Hybrid OLAP (HOLAP) tries to combine the advantages of relational and multi-dimensional database technology. The relational model is used to store replicated and low-level aggregates, while the multi-dimensional model is responsible for high-level aggregates.

Data Cube Operators

Data cubes are used for analytical purposes and not for (simple) transaction processing. Therefore there does not exist a clear boundary between data extraction or retrieval and data analysis. Therefore there does not exist a minimal, closed and complete set of OLAP operators. The mostly built-in operators on data cubes in commercial DWs are the following, see Shoshani (1997) and Jarke et al. (2000).

Slicing $\sigma_c(T)$ is to select data from a cube T according to a fixed condition c . This operation is called in Statistics conditioning if only frequencies (counts) applied to multi-way tables are considered. For example, we can retrieve data from `total_sales` according to $\sigma_{\text{product_id, customer_id, month, year==97}}(\text{total_sales})$.

Dicing $\pi_c(T)$ is table projection on T by selecting a sub-cube T' of some lower dimension c than the original cube T has. This operation is equivalent to marginalization in Statistics, i.e. projection of a data space into a lower dimension. For instance $\pi_{\text{date, customer_no}}(\text{total_sales})$ retrieves a sub-cube of total sales cross-classified by `date` and `customer`.

Table aggregation (roll-up) and *disaggregation* (drill-down) are operations on data cubes if at least on dimension is hierarchical. For example $\rho_{\text{year, customer_no, product_no}}(\text{total_sales})$ is a query for less fine-grained data, i.e. for `years` and summarizing over all months per year. This specific operation is called temporal aggregation. We observe that such an operation is not allowable if a type conflict happens with respect to the summary attribute. This is the case if the attribute 'sales' is substituted by 'no of employees', see Lenz and Shoshani (1997).

Drill-across $\delta_{\text{level, node, attribute}}(T)$ is a navigation on the same level through the various subtrees of a hierarchical attribute starting at a given node. For example, retrieving products from level 1 (`product_group`) with start at `product_group1` (shoes and balls) of the taxonomy "Product" delivers data about tennis nets.

In order to compute ratios, products etc. of data cubes the *join operator* $\gamma_{\otimes}(T_1, T_2)$ is needed. For instance, as `sales = turnover * price` we have `sales := $\gamma^*(\text{turnover}, \text{price})$` .

We note that there exist further operators like *pivot* (rotation of a cube), see Jarke et al. (2000), or *cube*, which was introduced by Gray et al. (1996). It delivers the margins *ALL* for any subset of dimensions.

9.3.4 Summarizability and Normal Forms

The main objective of *summarizability* is to guarantee correct results of the cube operation roll-up and the utilization of statistical (aggregation) functions like `min`, `max`, `avg`, `sum` and `count` under all circumstances, see Lenz and Shoshani (1997).

The corresponding integrity constraints are non-overlapping levels of dimensions, completeness and type compatibility. The first condition assures that each node of a taxonomy has at most one preceding node except for the root node. The second one ascertains that any node on a low level granularity corresponds to at least one node of a higher granularity. Type compatibility guarantees that the application of any statistical function on a summary attribute is sound. In a preceding section we mentioned the unfeasibility of aggregation of stocks over time. Another example is the misuse of the sum operator applied to code numbers of professions.

As Lehner et al. (1998) pointed out, the integrity constraint of completeness may turn out to be too restrictive. This happens if there exist structural missing values (null values) in taxonomies. For example, the German state Bavaria is divided into regions called “Kreise”. Berlin is a city as well as an autonomous German state. It is not divided into regions, but into suburbs called “Bezirke”. In such cases a context sensitive summarizability constraint is appropriate. The authors consequently proposed three multi-dimensional normal forms for fact tables. Lechtenböcker and Vossen (2001) improved the design of these normal forms.

Comparison of Terminologies

9.3.5

To sum up this chapter, Tables 9.3 and 9.4 compare the terminology of statistical databases and OLAP, see Shoshani (1997).

Table 9.3. Comparison of concepts

Statistical databases	OLAP
Categorical attribute	Dimension
Structural attribute	Dimension hierarchy
Category	Dimension value
Summary attribute	Fact
Statistical object, multidimensional table	Data cube
Cross product	Multidimensionality

Table 9.4. Comparison of operators

Statistical databases	OLAP
Table projection	Dice
Table selection	Slice
Table aggregation	Roll-up
Table disaggregation	Drill-down
Table join	term missing
Term missing	Drill across
Viewing	pivoting

9.4 Access Methods

9.4.1 Views (Virtual Tables)

Statistical databases are often accessed by different users with different intentions and different access rights. As already indicated in Sect. 9.2.2, these different requirements can be accounted for by using *views*. These views are derived *virtual tables*, which are computed from the (actually stored) base tables, see Elmasri and Navathe (1999). There are two main purposes for the use of views.

1. It makes the use of the DBS or DW more convenient for the user by providing only customized parts of the whole data cube.
2. It enforces security constraints by restricting operations on the base tables and by granting users access to their specific views only.

The following SQL statement creates a view for the manager of the product “Tennis Nets” from our example in Table 9.1. It only permits to look up the revenues for “Tennis Nets” while for all other products, viewing the sales and modifying the corresponding base tables is not possible.

```
CREATE VIEW tennis_nets_manager AS
  SELECT date.month, date.year, customer_id,
         sum(sales)
  FROM total_sales WHERE product_name = "Tennis
  Nets" ;
```

Views can never contain information that is not present in the base tables as the DBS translates all view queries into equivalent queries that refer only to base tables.

Base tables of a DW may contain millions of tuples. Scanning these tables can be time-consuming and may slow down the interaction between the decision support system and the user significantly. One strategy to speed up the access to aggregated data is to pre-compute a range of probable queries and to store the results in *materialized views*, see Gupta et al. (1997). The access to these materialized views is then much faster than computing data on demand. Yet there are drawbacks to this strategy. The pre-computed data need space, the prediction of the users’ queries is difficult, and each change in the base table requires an update of the materialized view also. This is known as the *view maintenance problem*, see Huyn (1997).

9.4.2 Tree-based Indexing

The tables of a DW can physically be accessed either by a sequential scan or by random access. With today’s hard disks, a sequential scan is 10 to 20 times faster than random access, see Jürgens (2002). That means if more than approximately 5% to 10% of the data has to be accessed in a table, it is faster to scan the entire table than addressing specific tuples via random access. In order to avoid full table scans, the number of tuples involved in the result computation has to be reduced.

This can be achieved via *index structures*, which permit a fast look-up of specific tuples.

The best-known index structure for one-dimensional data (i.e. data with just one key such as `product_id`) is the *B-tree*, see Bayer and McCreight (1972), Comer (1979). Pointers to the data items are stored in the leaf nodes of a balanced tree. The B-tree is a very general and flexible index structure, yet in some specific cases it may be outperformed by different kinds of hashing, see Gaede and Günther (1998).

The *universal B-tree (UB-tree)*, see Bayer, 1997) is an extension of the B-tree for indexing multidimensional data such as `total_sales (date.month, date.year, customer_id, product_name, sum(sales))`. The approach partitions the multidimensional data space into squares each of which is captured by a space-filling Z-curve, see Fig. 9.6. For each record, the Z-address of the square, which contains the key values is computed. These Z-addresses are one-dimensional and serve as the new primary keys for the records, which can then be indexed with a standard B-tree.

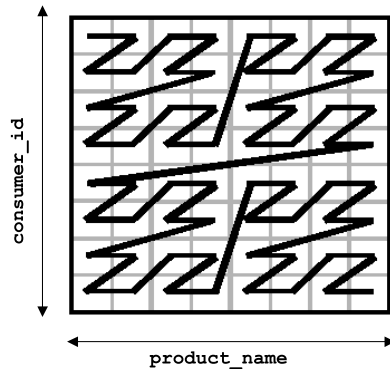


Figure 9.6. The UB-tree: partition and capture of multidimensional space with the Z-curve

Another approach for indexing multidimensional data is the *R-tree*, see Guttman (1984). It uses rectangles to represent multidimensional intervals. The leaf rectangles correspond to entries in the database. The parent nodes contain all child nodes and the minimal bounding rectangle. The root rectangle covers the entire query space. An example of how to store sales indexes in an R-tree when `product_name` and `customer_id` build the concatenated primary key is shown in Fig. 9.7. The minimal bounding rectangle of the dashed-line rectangles A, B and C constitutes the entire search space.

Refinements are the R^+ -tree of Sellis et al. (1985), the R^* -tree of Beckmann et al. (1990) and a slightly improved version called R_a^* -tree of Jürgens (2002).

Bitmap Index Structures

An important alternative to tree index structures is *bitmap indexing*. For each value of an attribute, a bitmap vector indicates whether or not it is assumed

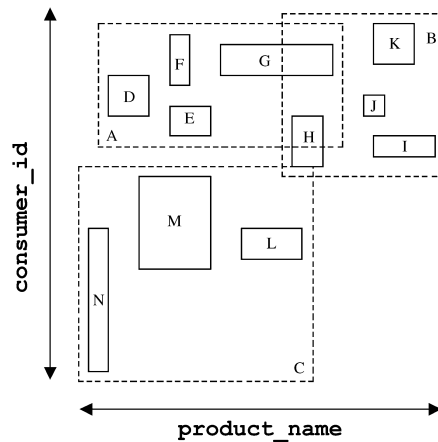


Figure 9.7. An exemplary R-tree

in the records of the table, see Chan and Ioannidis (1998), O’Neil and Quass (1997), Wu and Buchmann (1998). Table one shows a bitmap index for the attribute `product_name` corresponding to the example presented in Table 9.5.

Table 9.5. Bitmap index for the attribute `product_name`

transaction_id	Tennis balls	Tennis nets	Tennis shoes
015	0	0	1
018	1	0	0
004	0	1	0
009	0	0	1

The bitmap vector for the attribute value “Tennis Balls” is $(0, 1, 0, 0)^T$. Such a set of bitmap vectors is created for all dimensions. In our `total_sales` example, bitmap indexes have to be created further for `(date.month, date.year)` and `customer_id`.

The size of the bitmap index depends on the number of tuples and on the cardinality of the attribute domain. As the required operations on bitmaps are simple they are very fast. Thus loading blocks from disc and performing the basic Boolean operations is efficient, especially if the number of dimension is high, see Jürgens (2002). As bitmaps are often sparse, they are well suited for compression techniques. This is the reason why many commercial DBSs are implemented using bitmaps. However, standard bitmaps indexes become space consuming for high attribute’s domain cardinality, and they are not very efficient for (low dimensional) range queries, which are typical for DW systems.

Several approaches have been proposed to overcome these drawbacks like the multi-component equality encoded bitmap index, see Chan and Ioannidis (1998). The basic idea is to compress bitmap indexes by encoding all values into a smaller

number system by applying modular multiplication. This significantly reduces the space requirements for attributes of high cardinality.

To summarize, bitmaps are more suited for high-dimensional queries with low attribute cardinality. Tree index structures are better for low-dimensional range queries with attributes of high cardinality.

Extraction, Transformation and Loading (ETL)

9.5

ETL is a shorthand notation for a workflow of the initial popularization or a follow-up update of a DW, a data mart or an OLAP application. In the first step data must be extracted from the various data sources and temporarily stored in a so-called staging area of a DWS. Transformation means to modify data, schema and data quality according to requirement specifications of the DWS. Loading is the integration of replicated and aggregated data in the DW. As the data volume may be huge, incremental loading within pre-selected time slots by means of a bulk loader is appropriate.

Extraction

Extraction can be triggered by events linked to time and state of a DBS in operation or can be executed under human control. Mostly extraction is deferred according to an extraction schedule supplied by monitoring of the DWS. However, changes of data in the source system are tracked in real time, if the actuality of data is mandatory for some decision makers, see Kimball (1996).

As the data sources are generally heterogeneous, the efforts to wrap single data sources can be enormous. Therefore software companies defined standard interfaces, which are supported by almost all DBMS and ETL tools. For example, the OLE database provider for ODBC, see Microsoft (1998, 2003), Oracle (2003) and IBM (2003).

Transformation

Transformations are needed to resolve conflicts of schema and data integration and to improve data quality, see Chap. III.9.

We first turn to the first type of conflicts. Spaccapietra et al. (1992) consider four classes of conflicts of schema integration, which are to be resolved in each case.

1. *Semantic conflicts* exist, if two source schemas refer to the same object, but the corresponding set of attributes is not identical, i.e. the class extensions are different. As an example take two customer files. One record structure includes the attribute name gender, while it is missing in the other one.
2. A second kind of conflict of integration happens if *synonyms*, *homonyms*, different *data types*, *domains* or *measurements units* exist. For instance, think of the synonym part/article, a homonym like water/money pool, string/date

- as a domain, and Euro/USD. The ambiguity of our natural language becomes clear when one thinks of the meaning of “name” – family name, nickname, former family name, artist name, friar name, ...
3. *Schema heterogeneity conflicts* appear if the source schemas differ from the target schema of the DW. For example, sales and departments can be modeled as two relations `Sales` and `Department` of a relational data model or as a nested relation `Department\Sales` as part of an object oriented model. Another kind of conflict corresponds to the mapping of local source keys to global surrogates, see Bauer and Günzel (2001). This problem gets tightened if entity identification is necessary in order to decide whether a pair of records from two data sources refer to same entity or not. Fellegi and Sunter (1969) were the first to solve this problem by the record-linkage technique, which is now considered as a special classification method, see Neiling (2003).
 4. *Structural conflicts* are present if the representation of an object is different in two schemas. There may exist only one customer schema with the attribute `gender` in order to discriminate between “males” and “females”. Alternatively, there may be two schemas in use, one linked to “females”, the other one to “males”.

The second type of conflicts, i.e., conflicts of data integration, happens, if false or differently represented data are to be integrated. False data are generated by erroneous or obsolete entries. Differences in representation are caused by non- identical coding like `male/female` versus `0/1` or by different sizes of rounding-off errors.

9.6 Metadata and XML

McCarthy (1982) described *metadata* as data about data. However, the technical progress of OLTP and OLAP DBSs, workflow techniques and information dissemination has made it necessary, to use a more general definition of metadata.

Metadata is now interpreted as any kind of integrated data used for the design, implementation and usage of an information system. This implies that metadata not only describes real data, but functions or methods, data suppliers or sources and data receivers or sinks, too. It does not only give background information about the technology of a DBS or DWS, but about its semantic, structure, statistics and functionality. Especially, the semantic metadata enable the common user to retrieve definitions of an attribute, to select and filter values of meta attributes, and to navigate through taxonomies.

In Fig. 9.8 we present a view of a conceptually designed metadata. Its core is given by a statistical object, which is either a specialisation of a data matrix or a data cube. It is uniquely described by a definition, and is related in a many to many way to validation and processing rules, surveys or reports and attributes. As we present only a view, no further refinement is given with respect to attributes like roles (measure, key, property), scales (nominal, ordinal, cardinal), ontologies

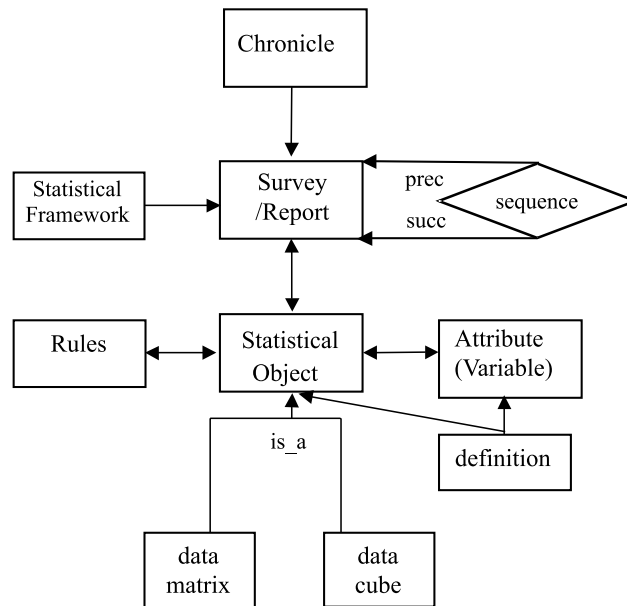


Figure 9.8. Statistical view of metadata

or even domains (natural, coded) etc. Each statistical object is linked to at least one survey or report. Surveys or reports can be sequenced according to preceding or succeeding ones, are related to a statistical framework (“statistical documentation”) giving details about sampling scheme and frame, population and statistical methods, and are associated to a chronicle as calendar of events. Furthermore references to the specific literature and law are included. The corresponding substructure is not displayed in Fig. 9.8. For further information about the metadata structure from the user’s point of view, see Lenz (1994).

As metadata is stored and can be retrieved similar to real data, it is captured in a *repository* and is managed by a metadata manager. A repository can be accessed by users, administrators and software engineers according to their privileges and read-write rights.

Such repositories are offered from various vendors. Microsoft (2001) labelled its repository as “metadata services”, and it is integrated in its SQL server. Alliances were founded to harmonize the metadata models and to standardize the exchange formats. Leading examples are the Open Information Model of the Metadata Coalition (MDC), see <http://www.mdcinfo.com>, and the Common Warehouse Metamodel (CWM), which was developed by the Object Management Group (OMG), see <http://www.omg.org>. Since the year 2000 both groups were fused and try to merge their models. Due to the increasing importance of XML and XML databases, import and export format of metadata based on XML is becoming an industrial standard. This happened to OLAP client-server architectures, see “XML for Analysis” as referred in Sect. 9.3.3.

9.7 Privacy and Security

9.7.1 Preventing Disclosure of Confidential Information

The statistical databases that are built by government agencies and non-profit organizations often contain confidential information such as income, credit ratings, type of disease or test scores of individuals. In corporate DWs, some strategic figures that are not related to individuals like sales for recently launched products may also be confidential. Whenever sensitive data is exchanged, it must be transmitted over a secure channel like the *Secure Socket Layer* (SSL), see Netscape (1996) in order to prevent unauthorized use of the system. For the purposes of this chapter, we assume that adequate measures for security and access control are in place, see Stallings (1999).

However, even if the information in the statistical database safely reaches the correct addressee, the system has to ensure that the released information does not compromise the privacy of individuals or other confidential information. Privacy breaches do not only occur as obvious disclosures of individual values in single queries. Often, the combination of multiple non-confidential query results may allow for the *inference* of new confidential facts that were formerly unknown.

We give an example. From Table 9.1, we take the total sales for “Tennis Shoes” (28,500), “Tennis Balls” (450), “Tennis Nets” (9450) and a fourth, new product (“Tennis Socks”, 500). We assume that sum queries for groups of products are allowed but that single, product-specific sales values are confidential. After querying the sum for balls and shoes (28,950) and for balls and socks (950), the user can infer an interval of [28,000; 28,950] for the sales of shoes, as sales cannot be negative. The length of the interval, which is the maximum error of the user’s estimation of the confidential shoe sales is only 3.3% of the actual value. This particular case of disclosure is called *interval inference*, see Li et al. (2002). Other types of inference include *exact inference* (concluding the exact value of 28,500 for shoes sales) and *statistical inference* (inferring estimates like mean

$$\bar{x}_{\text{Tennis Shoes}} = 30,000 \quad \text{and standard deviation} \quad s_{\text{Tennis Shoes}} = 5000 .$$

If a researcher is granted ad-hoc access to a statistical database, there are basically two different approaches to protect information that is private and confidential from being revealed by a malevolent snooper, see Adam and Wortmann (1989), Agrawal and Srikant (2000), Fig. 9.9. In the first approach, the kind and number of queries that a researcher poses to the statistical database is restricted (*query restriction*). In the second approach, the entire database is subject to a manipulation that protects single values but preserves the statistical properties which are of interest to the user. Then the perturbed database can be accessed by a researcher without restrictions (*data perturbation*). In the following, we give an overview of disclosure protection techniques of this kind.

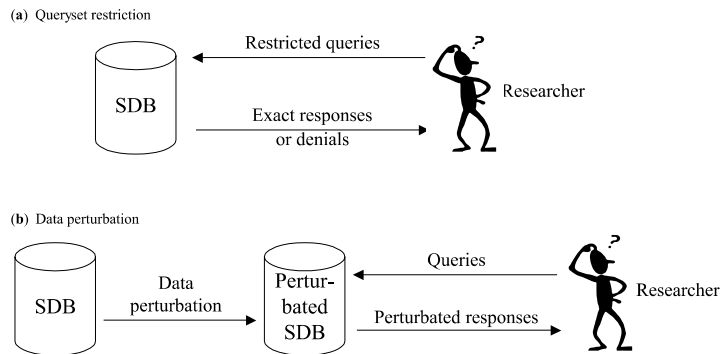


Figure 9.9. (a) Query set restriction and (b) data perturbation. Source: Adam and Wortmann (1989)

9.7.2 Query Set Restriction

With this approach a query is either denied or responded with an exact answer as the upper sketch in Fig. 9.9 indicates.

Query Set Size Control. Fellegi (1972) works by setting lower and upper bounds for the size of the query answer set based on the properties of the database and on the preferences fixed by the database administrator. If the number of returned records did not lie within these bounds, the information request would have to be rejected and the query answer is denied. As queries that are issued sequentially by one user often have a large numbers of entities in common, an improvement is the restriction of these entities to a maximum number, see Dobkin et al. (1979). Although popular, this method is not robust enough as a stand-alone solution, see Denning (1982).

Auditing. It involves keeping up-to-date logs of all queries made by each user and constantly checking for possible disclosures whenever a new query is issued. One major drawback of this method is that it requires huge amounts of storage and CPU time to keep these logs updated. A well-known implementation of such an audit system is *Audit Expert* by Chin and Özsoyoglu (1982). It uses binary matrices, see bitmap indexes in Sect. 9.4.3, to indicate whether or not a record was involved in a query.

Cell Suppression. See Cox (1980) is an important method for categorical databases when information is published in tabular form. Especially Census Bureaus often make use of tabular data and publish counts of individuals based on different categories. One of the main privacy objectives is to avoid answers of small size. For example, if a snooper knows somebody's residence, age and employer, he can issue a query for (ZIP = 10178, Age = 57, Employer = 'ABC'). If the answer is one entity, the snooper could go on and query for (ZIP = 10178, Age = 57, Employer = 'ABC', Diagnosis =

'Depression'). If the answer is one again, the database is compromised and the person with the diagnosis identified. The cells should have to be suppressed. A common criterion to decide whether or not to suppress a cell is the *N-k rule* where a cell is suppressed if the top N respondents contribute at least $k\%$ of the cell total. N and k are parameters that are fixed by the database administrator, i.e. the Census Bureau. In the exemplary case of $N = 2$ and $k = 10\%$, a cell which indicates aggregated income (\$10M) of 100 individuals would have to be suppressed if the top two earners' aggregate income exceeded \$1M.

9.7.3 Data Perturbation

In the query restriction approach, either exact data is delivered from the original database or the query is denied. As depicted in the lower sketch of Fig. 9.9, an alternative is to perturb the original values such that confidential, individual data become useless for a snooper while the statistical properties of the attribute are preserved. The manipulated data is stored in a second database and is then freely accessible for the users.

If in Table 9.1, we permute the sales of tennis balls, tennis nets and tennis shoes, individual sales data is not correct anymore. But the arithmetic average and the standard deviation of the attribute `sales` stay the same. This procedure is called *data swapping*, see Denning (1982).

Noise addition for numerical attributes, see Traub et al. (1984), means adding a disturbing term to each value: $Y_k = X_k + e_k$, where X_k is the original value and e_k adheres to a given probability distribution with mean zero. As for every value X_k value, the perturbation e_k is fixed, conducting multiple queries does not refine the snooper's search for confidential single values.

A hybrid approach are *random-sample queries*, Denning (1982), where a sample is drawn from the query set in such a way that each entity of the complete set is included in the sample with probability P . If, for example, the sample of a COUNT query has n entities, then the size of the not perturbed query set can be estimated as n/P . If P is large, there should be a set-size restriction to avoid small query sets where all entities are included.

9.7.4 Disclosure Risk versus Data Utility

All methods presented in the preceding sections aim at lowering the disclosure risk for data that is private and confidential. But at the same time, each of these methods reduces, in some way, the utility of the data for the legitimate data user. Duncan and Keller-McNulty (2001) present a formal framework to measure this trade-off between disclosure risk and data utility, the *Risk-Utility* (R-U) map. There are numberless measures for disclosure risk, see Domingo-Ferrer et al. (2002) for an excellent overview. We already gave an intuitive measure for interval inference. The sales for tennis shoes were predicted with an error of only 3.3%, see Sect. 9.7.1.

However, it is far more difficult to measure data utility because it strongly depends on the varying preferences of the data user. Especially for this reason,

classifying statistical disclosure control methods as presented here on an absolute scale is almost an impossible task.

References

- Adam, N. and Wortmann, J. (1989). Security-Control Methods for Statistical Databases: A Comparative Study. *ACM Computing Surveys*, 21(4):515–556.
- Agrawal, R. and Srikant, R. (2000). Privacy-preserving Data Mining. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pp. 439–450.
- Bauer, A. and Günzel, H. (eds) (2001). *Data Warehouse Systeme*. dpunkt.verlag, Heidelberg.
- Bayer, R. (1997). The universal B-Tree for multidimensional Indexing: General Concepts. In *World-Wide Computing and Its Applications '97 (WWCA '97)*. Tsukuba, Japan, 10–11, Lecture Notes on Computer Science, Springer Verlag.
- Bayer, R. and McCreight, E. (1972). Organization and maintenance of large ordered indexes. *Acta Informatica*, 1(3):173–189.
- Beckmann, N., Kriegel, H.-P., Schneider, R. and Seeger, B. (1990). The R*-tree: An efficient and robust access method for points and rectangles. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pp. 323–331, New York.
- Chan, C.-Y. and Ionanidis, Y.E. (1998). Bitmap index design and evaluation. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pp. 355–366.
- Chin, F.Y. and Özsoyoglu, G. (1982). Auditing and inference control in statistical databases. *IEEE Transactions on Software Engineering*, 8(6):574–582.
- Comer, D. (1979). The ubiquitous B-Tree. *ACM Computing Surveys*, 11(2):121–138.
- Cox, L.H. (1980). Suppression methodology and statistical disclosure control. *Journal of the American Statistical Association*, 75:377–385.
- Denning, D.E. (1982). *Cryptography and Data Security*. Addison-Wesley.
- Dobkin, D., Jones, A.K. and Lipton, R.J. (1979). Secure databases: Protection against user influence. *ACM Transactions on Database Systems*, 4(1):97–106.
- Domingo-Ferrer, J., Oganian, A. and Torra, V. (2002). Information-Theoretic Disclosure Risk Measures in Statistical Disclosure Control of Tabular Data. In *Proceedings of the 14th International Conference on Scientific and Statistical Database Management (SSDBM '02)*.
- Duncan, G. and Keller-McNulty, S. (2001). Disclosure risk vs. data utility: The R-U confidentiality map. *Technical Report. Statistical Sciences Group*. Los Alamos National Laboratory.
- Dzeroski, S. and Lavrac, N. (eds) (2001). *Relational Data Mining*, Springer Verlag, Heidelberg.
- Elmasri, R. and Navathe, S.B. (1999). *Fundamentals of Database Systems*, 3rd Edition, Addison Wesley.

- Fellegi, I.P. (1972). On the question of statistical confidentiality. *Journal of the American Statistical Association*, 67(337):7–18.
- Fellegi, I.P. and Sunter, A.B. (1969). A Theory of Record Linkage. *Journal of the American Statistical Association*, 40, 1183–1210.
- Gaede, V. and Günther, O. (1998). Multidimensional Access Methods. *ACM Computing Surveys*, 30(2):170–231.
- Gray, J., Bosworth, A., Layman, A. and Pirahesh, H. (1996). Data Cube: A Relational Aggregation Operator Generalizing Group-By, Cross-Tab, and Sub-Total. In *Proceedings of the 12th International Conference on Data Engineering (ICDE'96)*, New Orleans, USA, IEEE Computer Society, pp. 29–53.
- Gupta, H., Harinarayan, V., Rajaraman, A.J. and Ullman, D. (1997). Index selection for OLAP. In *Proceedings of the Thirteenth International Conference on Data Engineering (ICDE '97)*, Birmingham U.K., IEEE Computer Society.
- Guttman, A. (1984). R-trees: A dynamic index structure for spatial searching. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pp. 47–57.
- Huyn, N. (1997). Multiple-view self-maintenance in data warehousing environments. In *Proceedings of the 23rd Conference on Very Large Databases (VLDB)*, pp. 26–35.
- IBM (2003). DB2 OLAP Server. <http://www-3.ibm.com/software/data/db2/db2olap/library.html>
- Inmon, W.H. (1992). *Building the Data Warehouse*, Wiley, New York.
- Jarke, M. Lenzerini, M. Vassiliou, Y. and Vassiliadis, P. (2003). *Fundamentals of Data Warehouses*, 2nd ed., Springer, Berlin et al.
- Jürgens, M. (2002). *Index Structures for Data Warehouses*. Springer Lecture Notes in Computer Science, Berlin/Heidelberg/New York.
- Kimball, R. (1996). *The Data Warehouse Toolkit*. Wiley, New York.
- Lechtenbörger, J. and Vossen, G. (2001). Quality-oriented Data Warehouse Schema Design. In *information technology*, vol. 45, 190–195.
- Lehner, W., Albrecht, J. and Wedekind, H. (1998). Normal Forms for Multidimensional Databases. In *Proceedings of the 10th International Conference on Scientific and Statistical Data Management (SSDBM'98)*, Capri, Italia, 63–72.
- Lenz, H.-J. (1994). A rigorous treatment of Microdata, Macrodata and Metadata. In Dutter, R. (ed.), *Proceedings in Computational Statistics*, Physica, Heidelberg.
- Lenz, H.-J. and Shoshani, A. (1997). Summarizability in OLAP and Statistical Databases, In *Proceedings of the 10th International Conference on Scientific and Statistical Data Management (SSDBM'97)*, Washington, USA.
- Lenz, H.-J. and Thalheim, B. (2001). OLAP Databases and Aggregation Functions. In *Proceedings of the 14th International Conference on Scientific and Statistical Data Management (SSDBM'01)*, Washington, USA.
- Li, Y., Wang, L., Wang, X. and Jajodia, S. (2002). Auditing interval-based inference. In *Proceedings of the 14th Conference on Advanced Information Systems Engineering (CAiSE'02)*, Toronto, Canada.

- McCarthy, J. (1982). Metadata Management for Large Statistical Databases, In *Proceedings of the 8th International Conference on Very Large Data Bases*, Mexico City, Mexico.
- Messerschmidt, H. and Schweinsberg, K. (2003). *OLAP mit dem SQL-Server*, dpunkt. verlag, Heidelberg.
- Microsoft (1998). OLE DB for OLAP Programmer's Reference.
- Microsoft (2003). Microsoft SQL Server: Data Transformation Services (DTS). <http://www.microsoft.com/sql/evaluation/features/datatran.asp>
- O'Neil, P. and Quass, D. (1997). Improved query performance with variant indexes. *SIGMOD record*, 26(2):38–49.
- Oracle (2003). Oracle Warehouse Builder – Product Information. <http://otn.oracle.com/products/warehouse/index.html>
- Neiling, M. (2003). *Identifizierung von Realwelt-Objekten in multiplen Datenbanken*. PhD dissertation, Univ. of Cottbus, Germany.
- Netscape (1996). Secure Socket Layer 3.0 Specification <http://wp.netscape.com/eng/ssl3/>
- Raden, N. (1996). Star Schema 101. Archer Decision Sciences, Santa Barbara, http://members.aol.com/nraden/str101_e.htm (2000.12.12)
- Shoshani, A. (1997). OLAP and Statistical Databases: Similarities and Differences. In *Proceedings of the 16th ACM SIGACT-SIGMOD-SIGART Symposium on Principles and Database Systems (PODS'97)*, Tucson, USA, 185–196.
- Spaccapietra, S., Parent, C. and Dupont, Y. (1992). Model independent assertions for integration of heterogeneous schemas. *VLDB Journal*, 1(1), 81–126.
- Stallings, W. (1999). *Cryptography and Network Security, Principles and Practice*. Addison Wesley.
- Traub, J.F., Yemini, Y. and Wozniakowski H. (1984). The statistical security of a statistical database. *ACM Transactions on Database Systems*, 9(4):672–679.
- Wirth, N. (1986). *Algorithms and data structures*, Englewood Cliffs, Prentice Hall.
- Wrobel, S. (2001). Inductive Logic Programming for Knowledge Discovery in Databases. In Dzeroski, S. and Lavrac, N. (eds), *Relational Data Mining*, Springer Verlag, Heidelberg.
- Wu, M.-C. and Buchmann, A.P. (1998). Encoded bitmap indexing for data warehouses. In *Proceedings of the 14th International Conference on Data Engineering (ICDE)*, 220–230.

Interactive and Dynamic Graphics

II.10

Jürgen Symanzik

10.1	<i>Introduction</i>	294
10.2	<i>Early Developments and Software</i>	295
10.3	<i>Concepts of Interactive and Dynamic Graphics</i>	297
	Scatterplots and Scatterplot Matrices	297
	Brushing and Linked Brushing/Linked Views	299
	Focusing, Zooming, Panning, Slicing, Rescaling, and Reformatting	300
	Rotations and Projections	301
	Grand Tour	302
	Parallel Coordinate Plots	302
	Projection Pursuit and Projection Pursuit Guided Tours	304
	Pixel or Image Grand Tours	304
	Andrews Plots	305
	Density Plots, Binning, and Brushing with Hue and Saturation	305
	Interactive and Dynamic Graphics for Categorical Data	306
10.4	<i>Graphical Software</i>	306
	REGARD, MANET, and Mondrian	308
	HyperVision, ExplorN, and CrystalVision	309
	Data Viewer, XGobi, and GGobi	310
	Other Graphical Software	312
10.5	<i>Interactive 3D Graphics</i>	312
	Anaglyphs	312
	Virtual Reality	313
10.6	<i>Applications in Geography, Medicine, and Environmental Sciences</i>	316
	Geographic Brushing and Exploratory Spatial Data Analysis	316

Interactive Micromaps	318
Conditioned Choropleth Maps	320
10.7 Outlook	321
Limitations of Graphics	321
Future Developments	321

Introduction

Interactive and dynamic statistical graphics enable data analysts in all fields to carry out visual investigations leading to insights into relationships in complex data. Interactive and dynamic statistical graphics involve methods for viewing data in the form of point clouds or modeled surfaces. Higher-dimensional data can be projected into one-, two- or three-dimensional planes in a set of multiple views or as a continuous sequence of views which constitutes motion through the higher-dimensional space containing the data.

Strictly, there is some difference between interactive graphics and dynamic graphics. When speaking of interactive graphics only, we usually mean that a user actively interacts with, i.e., manipulates, the visible graphics by input devices such as keyboard, mouse, or others and makes changes based on the visible result. When speaking of dynamic graphics only, we usually mean that the visible graphics change on the computer screen without further user interaction. An example for interactive graphics might be the selection of interval lengths and starting points when trying to construct an optimal histogram while looking at previous histograms. An example for dynamic graphics might be an indefinitely long grand tour with no user interaction. Typically, interactive graphics and dynamic graphics are closely related and we will not make any further distinction among the two here and just speak of interactive and dynamic statistical graphics.

Two terms closely related to interactive and dynamic statistical graphics are exploratory data analysis (EDA) and visual data mining (VDM).

EDA, as defined by Tukey (1977), “*is detective work – numerical detective work – or counting detective work – or graphical detective work.*” Modern techniques and software in EDA, based on interactive and dynamic statistical graphics, are a continuation of Tukey’s idea to use graphics to find structure, general concepts, unexpected behavior, etc. in data sets by looking at the data. To cite Tukey (1977) again, “*today, exploratory and confirmatory can – and should – proceed side by side.*” Interactive and dynamic statistical graphics should not replace common analytic and inferential statistical methods – they should rather extend these classical methods of data analysis.

Data mining (DM) itself (Witten and Frank, 2000; Klösgen and Zytkow, 2002), see also Chap. III.13, is a field whose scientific basis has only begun to be explored over the last few years. DM exists as a result of the convergence of several fields including data bases, statistics, and artificial intelligence. Friedman (1998) discusses the connection between DM and statistics in more details and Wegman (2000) provides a definition of DM that links it with EDA and graphics: “*Data mining is exploratory data analysis with little or no human interaction using computationally feasible techniques, i.e., the attempt to find interesting structure unknown a priori.*” Simultaneously with an increasing interest in DM there has been the evolution of computer graphics, especially in the area of virtual reality (VR). Within the statistics framework, the area of EDA has evolved into a more sophisticated

area of interactive and dynamic statistical graphics. Recently, DM has been combined with statistical graphics, resulting in VDM (Cox et al., 1997; Inselberg, 1998; Symanzik et al., 1999a; Macedo et al., 2000; Böhlen et al., 2003). However, there exist several different definitions of the term VDM. Soukop and Davidson (2002) dedicate less than one page to “*dynamic visualizations that allow user interaction*” in their book on VDM.

In this chapter we will provide a general overview on existing methods and software for interactive and dynamic graphics. We will also provide a snapshot of current developments that may become a standard in the near future but may also be quickly forgotten again. All sections are supported by an extensive list of references that will allow every reader from novice to expert to become more familiar with a particular concept of interactive and dynamic graphics. More specifically, in Sect. 10.2, we will discuss early developments and software related to interactive and dynamic graphics. In Sect. 10.3, we will discuss the main concepts and in Sect. 10.4 some software products related to interactive and dynamic graphics. Interactive 3D graphics will be discussed in Sect. 10.5 and applications of interactive and dynamic graphics in geography, medicine, and environmental sciences will be discussed in Sect. 10.6. We conclude this chapter with an outlook to possible future developments in Sect. 10.7.

All graphical displays throughout this chapter are based on the “Places” data set that was distributed to interested members of the American Statistical Association (ASA) several years ago so that they could apply contemporary data analytic methods to describe these data and then present results in a poster session at the ASA annual conference. The data are taken from the Places Rated Almanac (Boyer and Savageau, 1981). The data are reproduced on disk by kind permission of the publisher, and with the request that the copyright notice of Rand McNally, and the names of the authors appear in any paper or presentation using these data. The data consist of nine measures of livability for 329 cities in the U.S.: Climate and Terrain, Housing Cost, Health Care and Environment, Crime, Transportation, Education, The Arts, Recreation, and Economics. For all but two of the above criteria, the higher the score, the better. For Housing Cost and Crime, the lower the score the better. The scores are computed using several statistics for each criterion (see the Places Rated Almanac for details). Latitude and longitude have been added by Paul Tukey. Population numbers have been added as well.

10.2 Early Developments and Software

There is a strong history of statistical graphics research on developing tools for visualizing relationships between many variables. Much of this work is documented in videos available from the ASA Statistical Graphics Section Video Lending Library at <http://www.bell-labs.com/topic/societies/asagraphics/library/index.html>. Currently, these historical videos are upgraded into a digital format and will be available on CD or DVD later in 2004.

Additional material on statistical graphics can also be found in journals such as “Journal of Computational and Graphical Statistics”, “Computational Statistics”, and “Computational Statistics & Data Analysis” and in “Computing Science and Statistics”, the proceedings of the Interface conferences. The following paragraphs only serve as a basic overview for readers unfamiliar with dynamic statistical graphics. They are not intended as a full introduction into this topic.

A video clip of the successive stages in a multidimensional scaling algorithm (Kruskal, 1970) is one of the first examples how to apply dynamic statistical graphics. A second example by Chang (1970) shows an interactive search for a structured two-dimensional projection in five dimensions where three of the five dimensions are noise. PRIM-9 (Picturing, Rotation, Isolation and Masking in up to 9 dimensions), documented in Fisherkerler et al. (1974a) and Fisherkerler et al. (1974b), is the landmark example of early dynamic statistical graphics. Projections formed the fundamental part of the visualization system and were complemented with isolation and masking. A good explanation of the importance of projection as a tool for visualizing structure in high-dimensional data can be found in Furnas and Buja (1994).

One major breakthrough in using projections for visualizing higher dimensions was made by Asimov (1985) in his work on the grand tour. The grand tour, further exploited in Buja and Asimov (1986a), in an abstract sense shows a viewer all possible projections in a continuous stream (which could be considered to be moving planes through p -dimensional space). Several possibilities for “showing all possible projections” were explored in the original work, but the most successful method to arise from it is based on interpolating between random planes. Another common approach to displaying high-dimensional data can be found in Becker and Cleveland (1988) where data is plotted in a scatterplot matrix, i.e., a matrix of pairwise scatterplots. Users can do linked brushing among the plots, i.e., mark points with different symbols and colors, while this information is also immediately displayed in all related (linked) plots.

The historical development of interactive and dynamic statistical graphics is well documented in a series of books and articles. Chambers et al. (1983) and du Toit et al. (1986) can be placed somewhere inbetween Tukey’s original idea of EDA and the beginning of modern dynamic and interactive statistical graphics. Wegman and DePriest (1986) is a collection of papers presented at a workshop sponsored by the Office of Naval Research (ONR), held in Luray, Virginia, from 24 through 27 May, 1983. About half of the papers are related to statistical image processing while the other half is related to (interactive) statistical graphics. Cleveland and McGill (1988) contains a collection of papers about dynamic graphics for statistics, originally published between 1969 through 1988. This book is a very good reference to see the progress in dynamic graphics concepts and software over two decades, starting from the very early stages through the late 1980’s. Buja and Tukey (1991) is based on the proceedings of the Institute for Mathematics and its Applications (IMA) 1989 summer program on “Robustness, Diagnostics, Computing and Graphics in Statistics”. An earlier “Handbook of Statistics, Vol-

ume 9, Computational Statistics”, edited by Rao (1993), contains several then state-of-the-art overviews on interactive and dynamic statistical graphics, most notably the chapters by Wegman and Carr (1993) and Young et al. (1993). Nagel et al. (1996) dedicate two (out of six) chapters of their book to dynamic graphics – one being an overview and one discussing applications. Theus (1996) is fully dedicated to the theory and applications of interactive statistical graphics. Wilhelm et al. (1996) contains reviews of software for interactive statistical graphics.

Major statistical journals often dedicate special issues to interactive and dynamic graphics, e.g., “Computational Statistics” (Volume 14, Issue 1, 1999) on “Interactive Graphical Data Analysis” and “Computational Statistics & Data Analysis” (Volume 43, Number 4, 2003) on “Data Visualization”.

Concepts of Interactive and Dynamic Graphics

10.3

This section will provide some deeper insights into concepts of interactive and dynamic graphics mentioned in the previous sections. Buja et al. (1996) contains a taxonomy of interactive data visualization based on the notions of focusing, linking, and arranging views of data. Unwin (1999) discusses some of the main concepts in the context of interactive graphics software.

10.3.1 Scatterplots and Scatterplot Matrices

Perhaps the most basic concepts for statistical graphics are scatterplots (see Figs. 10.1, 10.2, 10.3, and 10.4). In a simple scatterplot, we place different symbols (sometimes also called glyphs) at x - and y -positions in a two-dimensional plot area. These positions are determined by two of the variables. The type, size, and color of the symbols may depend on additional variables. Usually, explanatory information such as axes, labels, legends, and titles are added to a scatterplot. Additional information such as a regression line or a smoothed curve can be added as well.

If the data consist of more than two variables (e.g., somewhere between three to ten), the data can be displayed by a scatterplot matrix (see Figs. 10.2 and 10.3) that shows all pairwise scatterplots of the variables. The essential property of a scatterplot matrix is that any adjacent pair of plots have one of their axes in common. When plotting the full array of all $n \times (n - 1)$ pairwise scatterplots, each plot in the upper triangle of plots has a matching plot in the lower triangle of plots, with the exception that the axes in these pairs of plots have been flipped. Therefore, sometimes only the upper or lower triangle of scatterplots is displayed; thus gaining plotting speed and allowing each individual plot to be somewhat larger. Early examples of scatterplot matrices can be found in Chambers et al. (1983) and Cleveland (1985) for example. Chambers et al. (1983) initially called

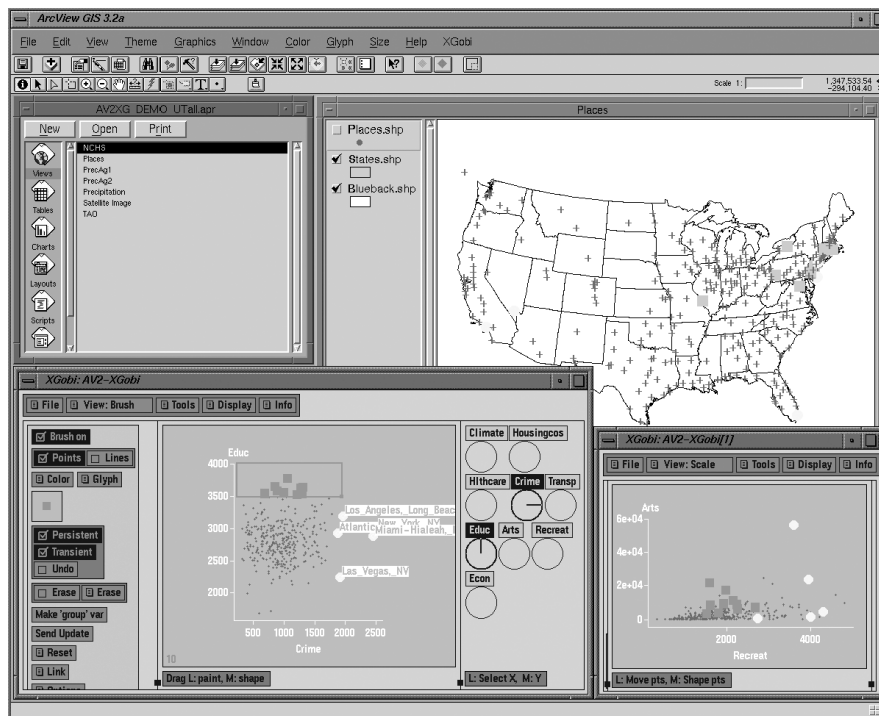


Figure 10.1. Screenshot of the “Places” data in ArcView/XGobi. A map view of the 329 spatial locations is displayed in ArcView at the top. The two XGobi windows at the bottom are showing scatterplots of Crime (horizontal) vs. Education (vertical) (*left*) and Recreation (horizontal) vs. Arts (vertical) (*right*). Locations of high Crime have been brushed and identified, representing some of the big cities in the U.S. Also, locations of high Education (above 3500) have been brushed, mostly representing locations in the northeastern U.S. All displays have been linked

an array of pairwise scatterplots for three variables a draftsman’s display and for four (or more) variables a generalized draftsman’s display. In their (generalized) draftsman’s display, each point is plotted with the same symbol. When encoding additional information through the use of different plotting symbols, Chambers et al. (1983) speak of symbolic (generalized) draftsman’s displays. Today, we hardly make any distinction of these different types of displays and just speak of scatterplot matrices.

Murdoch (2002) and Unwin (2002) discuss features good scatterplots and related interactive software should provide, e.g., meaningful axes and scales, features for rescaling and reformatting, good handling of overlapping points and missing data, panning and zooming, and querying of points. Carr et al. (1987) describe techniques for scatterplot matrices particularly useful for large numbers of observations.

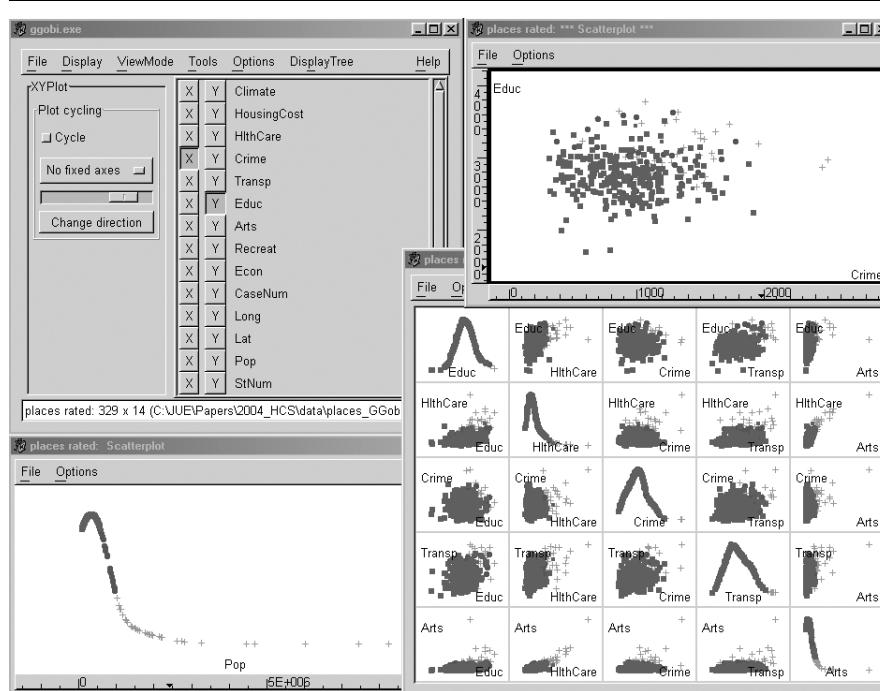


Figure 10.2. Screenshot of the “Places” data in GGobi. A scatterplot of Crime (horizontal) vs. Education (vertical) is displayed at the top right, a scatterplot matrix of five of the variables is displayed at the bottom right, and a density (1D) plot of Population is displayed at the bottom left. The data has been brushed with respect to Population: one group for a Population less than 500,000, one group for a Population between 500,000 and 1,000,000, and one group for a Population above 1,000,000. The scatterplot of Crime and Education seems to reveal that higher Population is associated with higher Crime and higher Education. The scatterplot matrix seems to reveal that higher Population is also associated with higher Arts and higher HealthCare. All displays have been linked

10.3.2 Brushing and Linked Brushing/Linked Views

Brushing, as introduced in Becker and Cleveland (1988) and Becker et al. (1988b), initially was considered as a collection of several dynamic graphical methods for analyzing data displayed in a scatterplot matrix. The central idea behind brushing is a brush, usually a rectangular area on the computer screen, that is moved by the data analyst to different positions on the scatterplot or any other graphical display. Four brushing operations were introduced in Becker and Cleveland (1988): highlight, shadow highlight, delete, and label. The most commonly used brushing technique is highlighting – often in the context of linked brushing, i.e., for linked views. All points that are inside the brush in the currently selected display are highlighted, i.e., marked with a different symbol or color. Simultaneously, points that correspond

to those points are automatically highlighted with the same symbol/color in all linked views.

A very useful brushing technique is the transient paint mode. As the brush is moved, the new points that come inside the brush are highlighted while points that move outside the brush are no longer highlighted.

While brushing initially was only developed for scatterplot matrices, it quickly has been adapted to other types of linked graphical displays. Linked brushing among different displays is one of the most useful techniques used within dynamic and statistical graphics. Linked brushing can be applied to graphical representation of continuous data, summary data such as histograms (Stuetzle, 1988), or even displays of categorical data such as mosaic plots (Hofmann, 2000, 2003). All dynamic statistical graphics software packages support linked brushing among different types of graphical displays these days.

When dealing with massive data sets, it is often beneficial to focus on particular subgroups of the data and also be able to quickly return to a previous stage of the analysis. Selection sequences (Theus et al., 1998; Hofmann and Theus, 1998) are an extension of the conventional linked-highlighting paradigm as they store the whole hierarchical path of a selection and allow an easy editing, redefinition, and interrogation of each selection in the path of the analysis. In a selection sequence, we can easily jump from one branch of the hierarchic selection tree to another.

Focusing, Zooming, Panning, Slicing, Rescaling, and Reformatting

10.3.3

Focusing techniques, as introduced in Buja et al. (1991), are based on the idea that it often might be easier for a human analyst to understand several individual displays, each focused on a particular aspect of the underlying data, rather than looking at the full data set. Focusing techniques include subset selection techniques, e.g., panning and zooming or slicing, and dimensionality reduction techniques, e.g., projection. Methods for focusing can be automatic, interactive, or a combination of both. While focusing shows only part of the data at a time, it is important to display multiple linked views of the data, perhaps each focusing on a different aspect of the data, to maintain the full picture of the data.

Zooming is a technique that can be used for inspecting details of the data when overplotting arises. Zooming can be done via some kind of a magnifying glass or by manually selecting subsections of the visible axes, e.g., via sliders. The main idea behind zooming is that when several points overplot in the full display, it may indeed turn out that these points are exactly the same when zooming into the neighborhood of these points – or, what most frequently happens, that these points have a particular structure and are not exactly the same.

Panning is closely related to zooming. An analyst should know which subset of the data is currently visible. Therefore, an information plot should reveal the current location on which subregion we have zoomed.

Slicing, as described in Furnas (1988) and Furnas and Buja (1994), is a technique that takes sections (or slices) of a high-dimensional data set. While slicing (and projections) are useful means for an exploratory data analysis, these techniques also have their limitations. However, these limitations may be overcome by combining slicing and projections in so-called projections (Furnas and Buja, 1994). An extension of individual projection views is the projection matrix (Tweedie and Spence, 1998), some kind of a density plot summarizing multi-dimensional volumetric information. The projection matrix is a useful representation for engineering design, allowing an analyst to interactively find a design that leads to a maximal manufacturing yield.

Rescaling is a technique that allows a user to quickly change the scale of the displayed variables, e.g., by taking the log, square root, standardize, or by mapping to a 0–1 scale. When looking at multiple variables, it might also be beneficial to have a common scale (from the minimum across all variables to the maximum across all variables). By interactively rescaling variables, an analyst may identify useful transformations for a follow-up modeling step of the data.

Reformatting includes features as simple as swapping x and y axes in a scatterplot or changing the order of coordinate axes in a parallel coordinate plot.

Unwin (2002) provides more details on several of the techniques described above.

10.3.4 Rotations and Projections

Rotation, as introduced in Fisher et al. (1974b) and later refined in Becker et al. (1988b), is a very powerful tool for understanding relationships among three or more variables. The familiar planar scatterplot is enhanced by rotation to give the illusion of a third dimension. We typically rotate plots in search of some interesting views that do not align with the plot axes and therefore cannot be seen in a scatterplot matrix. Usually, a three-dimensional point cloud representing three of the variables is shown rotating on a computer screen. The rotation shows us different views of the points and it produces a 3D effect while moving, allowing us to see depth. Basic rotation controls with a mouse have been introduced in Becker et al. (1988b).

Mathematically speaking, each rotation within a 3D space onto a 2D computer screen is based on a projection. Obviously, it is mathematically possible to project high-dimensional data onto low-dimensional subspaces and gain insights into the underlying data through dynamic visualizations of such projections. One particular example of a continuous sequence of projections, the grand tour, will be discussed in the next section. Cook and Buja (1997) discuss methods how to manually control high-dimensional data projections. Cook (1997) provides a variety of training data sets that help new users to get a visual feeling of the underlying high-dimensional data set when seen as a projection into low-dimensional space.

Grand Tour

10.3.5

Often, simple plot rotation, as discussed in the previous section, does not suffice to see all interesting views of the data. To produce a plethora of possible interesting views, the grand tour has been introduced in Asimov (1985) and Buja and Asimov (1986b). In Asimov (1985), the grand tour has been described as “a method for viewing multivariate statistical data via orthogonal projections onto a sequence of two-dimensional subspaces. The sequence of subspaces is chosen so that it is dense in the set of all two-dimensional subspaces.” Some of the features the grand tour can be used for are examining the overall structure and finding clusters or outliers in high-dimensional data sets.

In the context of the grand tour, an alternating sequence of brushing, looking at additional projections from the grand tour, brushing, and so on, is referred to as the brush-tour strategy in the remainder of this chapter. We can only be sure that a cluster visible in one projection of the grand tour really is a cluster if its points remain close to each other in a series of projections and these points move similarly when the grand tour is activated. If points move apart, we probably found several subclusters instead of one larger cluster.

Wegman (1992) discusses a form of the grand tour for general d -dimensional space. The algorithms for computing a grand tour are relatively computationally intensive. Wegman and Shen (1993) discuss an approximate one- and two-dimensional grand tour algorithm that was much more computationally efficient than the Asimov winding algorithm. That algorithm was motivated in part by a discussion of the Andrews (multidimensional data) plot, discussed in Sect. 10.3.9, which can also be regarded as a highly restricted pseudo tour.

Parallel Coordinate Plots

10.3.6

Parallel coordinate plots (Inselberg, 1985; Wegman, 1990)(see Fig. 10.3) are a geometric device for displaying points in high-dimensional spaces, in particular, for dimensions greater than three. The idea is to sacrifice orthogonal axes by drawing the axes parallel to each other resulting in a planar diagram where each d -dimensional point (x_1, \dots, x_d) is uniquely represented by a continuous line. The parallel coordinate representation enjoys some elegant duality properties with the usual Cartesian coordinates and allows interpretations of statistical data in a manner quite analogous to two-dimensional Cartesian scatterplots. This duality of lines in Cartesian plots and points in parallel coordinates extends to conic sections. This means that an ellipse in Cartesian coordinates maps into a hyperbola in parallel coordinates. Similarly, rotations in Cartesian coordinates become translations in parallel coordinates.

The individual parallel coordinate axes represent one-dimensional projections of the data. We can isolate clusters by looking for separation between data points on any axis or between any pair of axes. Because of the connectedness of the multidimensional parallel coordinate diagram, it is usually easy to see whether or not this clustering propagates through other dimensions.

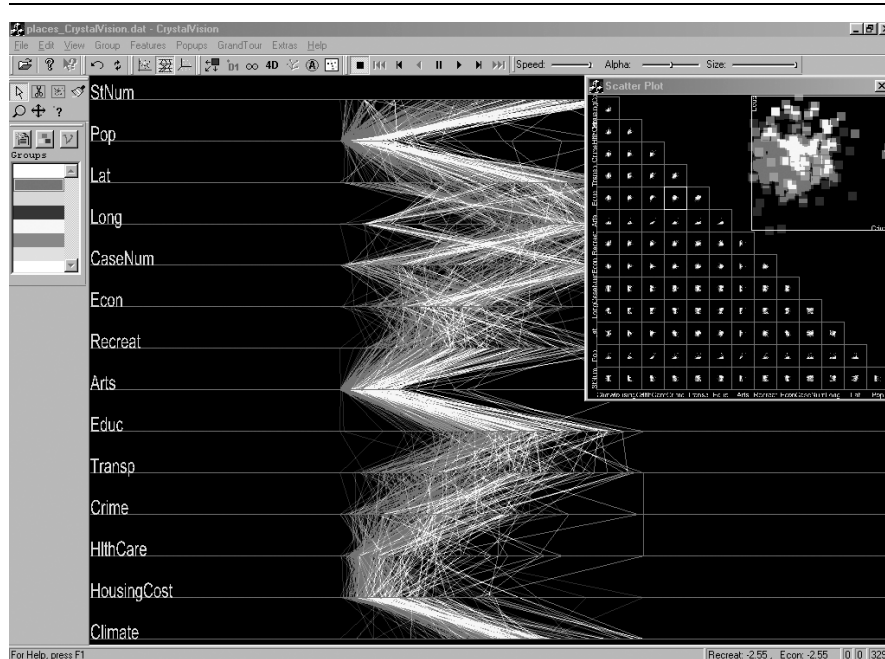


Figure 10.3. Screenshot of the “Places” data in CrystalVision. A parallel coordinate plot of all variables is shown as the main plot. A scatterplot matrix of all variables with a scatterplot of Crime (horizontal) vs. Education (vertical) is shown as a popup in the top right. The data has been brushed according to high and low Population. According to the parallel coordinate plot, higher Population is associated with higher Arts and HousingCost. The scatterplot of Crime and Education seems to reveal that higher Population is also associated with higher Crime and higher Education. All displays have been linked

The use of parallel coordinate plots for a d -dimensional grand tour sequence, sometimes called a parallel coordinate grand tour, has been described in Wegman (1992) and Wegman and Carr (1993). By using such a parallel coordinate grand tour, an analyst can find orientations where one or more clusters are evident. The general strategy for detecting clusters is the following: We begin with a static plot of the data in parallel coordinates. If there are any gaps along a horizontal axis (which incidentally does not need to coincide with the coordinate axes), then we color the individual clusters with distinct colors. Once all clusters are identified in the original coordinate system, we run the grand tour until an orientation of the axes is found in which another gap in one of the horizontal axes is found. Again we color the individual subclusters with distinct colors. This procedure is repeated until no further subclusters can be identified. This is another example of the brush-tour strategy referred to in Sect. 10.3.5. Indeed, when to stop is a matter of judgement, since the procedure can be repeated until practically every data point can be individually colored. The crucial issue, which really depends on the dynamic graphics, is to see that clusters identified in this manner track coherently

with the grand tour animation. That is, data points of the same color stay together as the grand tour rotation proceeds. If they do not, then there are likely to be substructures that can be identified through further grand tour exploration.

Slopes of parallel coordinate line segments can also be used to distinguish clusters. That is, if a group of line segments slopes, say, at 45° to the horizontal and another group slopes at, say, at 135° to the horizontal, then even though the lines fully overlap in both adjacent parallel coordinate axes and there is no horizontal gap, these sets of lines represent two distinct clusters of points. Fortunately, when such indication of clustering exists, the grand tour will also find an orientation of axes in which there is a horizontal gap. Thus the general strategy is to alternate color brushing of newly discovered clusters with grand tour rotations until no further clusters can be easily identified.

In some software packages, the parallel axes in a parallel coordinate plot are drawn as horizontal lines (e.g., in ExplorN) while in other software packages they are drawn as vertical lines (e.g., in XGobi). While it may be argued that this makes no difference from a mathematical point of view, the wider aspect ratio in the horizontal mode coupled with a more usual sense of plotting data along an abscissa rather than along the ordinate tends to allow for an easier human interpretation. Detailed interpretations are given in Wegman (1990).

Projection Pursuit and Projection Pursuit Guided Tours

10.3.7

While the grand tour, as discussed in Sect. 10.3.5, is a dynamic tool, projection pursuit (Kruskal, 1969; Friedman and Tukey, 1974; Huber, 1985), see also Chap. III.6, is a static tool. Projection pursuit results in a series of static plots of projections that are classified as “interesting” with respect to a particular projection pursuit index. Many projection pursuit indexes, e.g., the ones discussed in Jones and Sibson (1987), Friedman (1987), Hall (1989), Morton (1989, 1992), Cook et al. (1993), and Posse (1995), are based on the idea to search for the most non-normal projections. Usually, each projection pursuit index, a function of all possible projections of the data, results in many hills and valleys. Friedman (1987) suggests a projection pursuit algorithm that initially searches for relatively high values of the function and then starts derivative-based searches to find the global maximum.

The combination of grand tour and projection pursuit, called projection pursuit guided tour (Cook et al., 1995), helps to direct the grand tour towards “interesting” projections. This combination of the two methods into an interactive and dynamic framework not only shows the “interesting” projections but it maintains the motion so the user has a feeling how successive “interesting” projections have been obtained.

Pixel or Image Grand Tours

10.3.8

The idea of the pixel or image grand tour (IGT) evolved from an initial application of one-dimensional tours to image data. Multiple registered images can be

regarded as a multidimensional image in which each pixel location has a vector attached to it. For example, ordinary red, green, and blue (RGB) color images are vector-valued images. The basic idea of the image tour is to apply the same one-dimensional grand tour to each vector for all pixel locations in an image. This combines the vectors into a scalar function of time which can be rendered as a time-varying gray-scale image. The Wegman and Shen (1993) algorithm generalizes easily to two dimensions, so that an alternate approach to the IGT is to project the multidimensional vector into two dimensions and render the image as a false color image with two complementary colors such as red and cyan. It should be noted that red and cyan are complementary colors in the RGB color model used for most computer monitors whereas red and green are complementary colors in the conventional color model, introduced by the Commission Internationale de l'Éclairage (CIE) in 1931. A detailed comparison of these two and other color models can be found in Foley et al. (1990), Chap. 13. The initial discussion of the IGT was given by Wegman et al. (1998). Additional examples of the IGT can be found in Symanzik et al. (2002b).

Currently, the IGT software, written in C++ by Qiang Luo, is available for Silicon Graphics, Inc., (SGI) workstations. To obtain a fast rendering rate of large images, the software intensively uses SGI hardware features such as the α -channel hardware. There exists also a MATLAB version of the IGT written by Wendy Martinez. Both versions of the IGT software are not accessible through a Web site but can be obtained from the corresponding software developers.

10.3.9 Andrews Plots

The Andrews (multidimensional data) plot, as introduced in Andrews (1972) is based on a series of Fourier interpolations of the coordinates of multi-dimensional data points. Points that are close in some metric will tend to have similar Fourier interpolations and therefore will tend to cluster in the Andrews plot. Thus, the Andrews plot is an informative graphical tool most useful to detect clustering.

Ideas underlying the Andrews plot and the grand tour are quite similar. However, in contrast to the grand tour, the Andrews plot is a static plot while the grand tour is dynamic. Although dynamic renditions of the Andrews plot exist, and these sometimes also are (incorrectly) referred to as one-dimensional grand tour (Crawford and Fall, 1990), the Andrews plot is not a grand tour since it cannot sweep out all possible directions as pointed out in Wegman and Shen (1993). Three-dimensional generalizations of the Andrews plot and other pseudo grand tours have been introduced in Wegman and Shen (1993) as well.

10.3.10 Density Plots, Binning, and Brushing with Hue and Saturation

Carr et al. (1987) present techniques for visualizing data in scatterplots and scatterplot matrices when the data consists of a large number of observations, i.e., when

overplotting of points frequently occurs using standard techniques. A key idea to address in the visualization of a large number of observations is based on the estimation and representation of densities. For this purpose, the data is often binned into an $n \times n$ matrix for two-dimensional representation (or an $n \times n \times n$ matrix for three-dimensional representation). Possibilities to visualize the number of data points in each bin can be based on gray-scale (or color) density representations or by symbol area such as using differently sized hexagon symbols, where the area of the plot symbol is proportional to the count in each bin. Carr (1991) further extends these ideas and presents additional low-dimensional displays for data that consist of a large number of observations. Scott (1992) provides a general overview on techniques for density estimation, including averaged shifted histograms (ASH) and kernel density estimators, including possible visualization techniques via contour surfaces, (transparent) α -level contours, and contour shells. Further details on multivariate density estimation and visualization can be found in Chap. III.4.

Wegman and Luo (1997a) use hue and saturation for plotting and brushing. For each individual point, the hue is almost fully desaturated with black. When points are overplotted, the hue components are added. The saturation level should be interactively adjustable by the analyst. If many points overplot, the pixel will be fully saturated. If fewer points overplot, the pixel will be shown in a less saturated color. Often, computer hardware devices such as the α -channel allow the blending of pixel intensities with no speed penalties. When using saturation for parallel coordinate plots and the level of saturation corresponds with the degree of overplotting, this creates a kind of parallel coordinate density plot (Wegman and Luo, 1997a, 1997b).

Interactive and Dynamic Graphics for Categorical Data

10.3.11

Although categorical data are quite common in the real world, little research has been done for their analysis and visualization when compared to quantitative data. However, there exist useful interactive and dynamic graphics for categorical data (Ostermann and Nagel, 1993; Theus and Wilhelm, 1998). For example, brushing and linking of categorical data represented via bar charts and pie charts can be as useful as for quantitative data (Hummel, 1996). Modified bar charts where the same height is used for each category and the width is varied according to the number of counts are called spine plots (Hummel, 1996). When interactively highlighting a category of interest, spine plots allow the analyst to visually compare the proportions in the different subcategories by looking at the heights of the highlighted areas. Examples of interactive graphics for categorical data such as spine plots and interactive mosaic plots (see Fig. 10.4) can be found in Hofmann (2000, 2003). Valero-Mora et al. (2003) discuss spreadplots (and their implementation in ViSta), a method for laying out and simultaneously controlling graphics for categorical data.

Blasius and Greenacre (1998) present a collection of papers dealing with the visualization of categorical data. Main topics include graphics for visualization, correspondence analysis, multidimensional scaling and biplots, and visualization

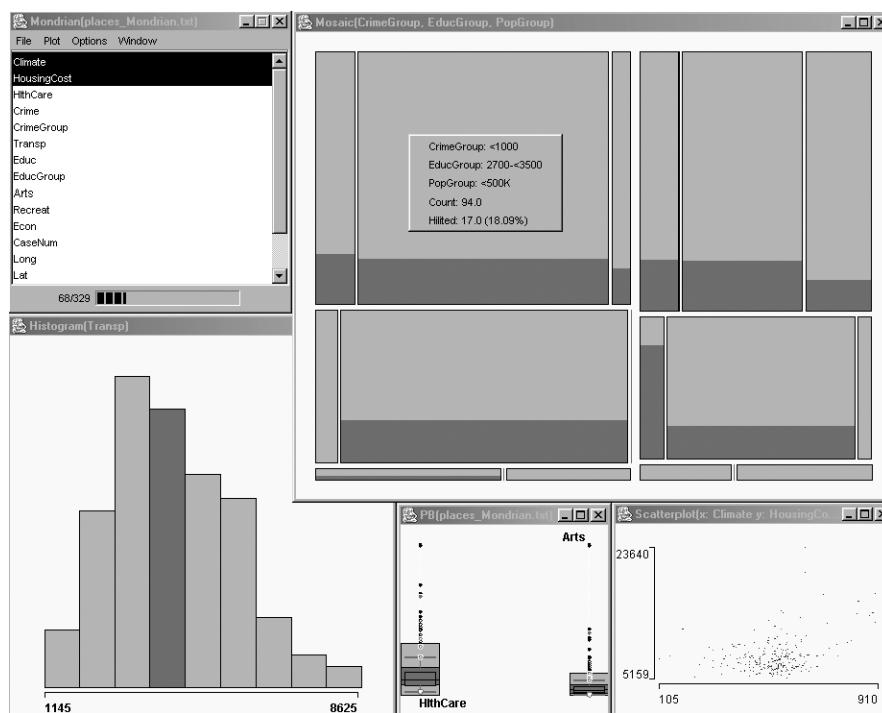


Figure 10.4. Screenshot of the “Places” data in Mondrian. The variables Crime, Education, and Population have been discretized for this figure. A mosaic plot of Crime (first vertical division, grouped as below 1000 (*left*) and above 1000 (*right*)), Education (first horizontal division, grouped as 2700 to 3500 (*top*), below 2700 (*middle*), and above 3500 (*bottom*)), and Population (second vertical division, grouped as 500,000 to 1,000,000 (*left*), below 500,000 (*middle*), and above 1,000,000 (*right*)) is displayed at the top right. A histogram of Transportation is shown at the *bottom left*, boxplots of HealthCare and Arts are shown at the *bottom middle*, and a scatterplot of Climate (horizontal) vs. HousingCost (vertical) is shown at the *bottom right*. The mosaic plot shows that Crime, Education, and Population are not independent. The different displays show how average Transportation (that has been brushed in the histogram) is related to the other variables. All displays have been linked

and modeling. Several of these approaches benefit from interactive and dynamic graphics.

10.4 Graphical Software

In this section, we concentrate on three main streams of software for interactive and dynamic statistical graphics: Software developed by researchers affiliated with the University of Augsburg, in particular REGARD, MANET, and Mondrian; soft-

ware developed by researchers affiliated with George Mason University (GMU), in particular ExplorN and CrystalVision; and software developed by researchers affiliated with Bell Labs, AT&T, and Iowa State University (ISU), in particular XGobi and GGobi. Wilhelm et al. (1996) contains an in depth review of software for interactive statistical graphics. Wilhelm et al. (1999) is one of the few publications where the different interactive graphical concepts provided by these three main streams (represented by MANET, ExplorN, and XGobi, respectively) are applied to the same data set and thus allow a direct comparison of their features and capabilities in visual clustering and classification.

REGARD, MANET, and Mondrian

10.4.1

In this section we present a series of software developments that was initiated in the late 1980's by John Haslett and Antony Unwin at Trinity College, Dublin, and later was continued by Antony Unwin and his collaborators at the Institut für Mathematik, University of Augsburg. Other main collaborators that contributed to the development of these software tools that should be mentioned here are Heike Hofmann, Martin Theus, Adalbert Wilhelm, and Graham Wills.

Some of the early developments are Diamond Fast (Unwin and Wills, 1988) and Spider (Craig et al., 1989). Diamond Fast is a software package for the exploration of multiple time series with interactive graphics. Spider is a software package for the exploration of spatially referenced data. Among its main features are moving statistics, an extension of brushing for spatial data (Craig et al., 1989). Spider also supports histograms, density estimates, scatterplot matrices, and linked brushing. It runs on Macintosh computers only.

REGARD (Unwin et al., 1990; Unwin, 1994) is a software package that also provides high interaction graphics tools for spatial data. REGARD stands for "Radical Effective Graphical Analysis of Regional Data" and runs on Macintosh computers only. REGARD supports four types of layers of spatial data, i.e., points, regions, lines, and pictures. The central display in REGARD is the map window that is linked to statistical displays such as boxplots, scatterplots, and rotating plots. A map may be loaded as one picture in a picture layer or as several pictures in several layers, thus allowing to turn on or off different aspects of a map (such as state boundaries or a road network). Additional interactive features are interrogation, highlighting, resizing, and rescaling. Advanced features include zooming into submaps, animation across ordered variables, cross-layer linking, network analysis tools, and interactive query tools across all graphical displays.

MANET (Unwin et al., 1996) is a statistical graphics research program for EDA and written in C++. MANET stands for "Missings Are Now Equally Treated" and runs on Macintosh computers only. It is freely available from the following Web site: <http://www1.math.uni-augsburg.de/Manet/>.

MANET offers all standard one- and two-dimensional graphics for continuous data as well as for discrete data: dotplots, scatterplots, histograms, boxplots, bar charts. Some special graphics for discrete and spatial data are integrated: spine plots, mosaic plots and polygon plots. MANET grew out of a project to keep track

of missing values in statistical graphics. In MANET all displays are fully linked and instantaneously updated. Displays are kept as simple as possible so they do not distract the user.

The standard use of linked views in MANET is to highlight clusters that are apparent in one dimension and to see these one-dimensional clusters in the light of other variables. By systematically subsetting the sample points, we can also detect two- and higher-dimensional clusters. Once a cluster has been detected, a classification rule can be set up by taking the boundary values of the cluster. In MANET those values can easily be obtained by interrogating the plot symbols.

One-dimensional views show the one-dimensional clusters directly. Two-dimensional clusters become visible by highlighting a subset in one variable and conditioning another plot on this subset. For three- and higher-dimensional clusters, we have to combine various subsets in different plots into one conditioning set and then we have to look at the remaining plots to check for clusters. The generation of such combined selections is not only possible in MANET but it is also very efficiently implemented through selection sequences.

In MANET, both dotplots and boxplots are drawn in a non-standard way. In dotplots the brightness of a point shows the frequency of its occurrence. This method, called tonal highlighting, is used to visualize overplotting of points. A bright color represents many points while a dark color represents just a few points. There is no tonal highlighting for selected points in MANET. The layout of boxplots is changed so that a standard boxplot can be superimposed for selected points. The inner fifty percent box is drawn as a dark grey box. The outer regions, usually represented as whiskers, are drawn as light grey boxes.

A recent new development, Mondrian (Theus, 2002, 2003), is a data visualization system written in JAVA and therefore runs on any hardware platform. Mondrian is freely available from the following Web site: <http://www.rosuda.org/Mondrian/>.

The main emphasis of Mondrian is on visualization techniques for categorical and geographical data. All plots in Mondrian (see Fig. 10.4) are fully linked and offer various interrogations. Any case selected in one plot in Mondrian is highlighted in all other linked plots. Currently, implemented plots comprise mosaic plots, scatterplots, maps, bar charts, boxplots, histograms, and parallel coordinate plots. Mosaic plots in Mondrian are fully interactive. This includes not only linking, highlighting and interrogations, but also an interactive graphical modeling technique for loglinear models.

10.4.2 **HyperVision, ExplorN, and CrystalVision**

In this section we present a series of software developments that was initiated in the late 1980's by Daniel B. Carr (initially with Battelle Pacific Northwest Laboratories) and Edward J. Wegman at GMU. Other main collaborators that contributed to the development of these software tools that should be mentioned here are Qiang Luo and Wesley L. Nicholson.

EXPLOR4 (Carr and Nicholson, 1988) is a research tool, originally implemented on a VAX 11/780 and written in FORTRAN. Its main features are rotation, masking, scatterplots and scatterplot matrix, ray glyph plots, and stereo views.

HyperVision, presented in Bolorforoush and Wegman (1988), is a software product that has been implemented in PASCAL on an IBM RT under the AIX operating system as well as for MS-DOS machines. The latter implementation has a mouse-driven painting capability and can do real-time rotations of 3D scatterplots. Other displays are parallel coordinate plots, parallel coordinate density plots, relative slope plots, and color histograms. The main interactive features in HyperVision in addition to linked brushing are highlighting, zooming, and nonlinear rescaling of each axis.

ExplorN (Carr et al., 1997) is a more advanced software package than HyperVision and EXPLOR4, but with similar basic features. It runs on SGI workstations only, using either the GL or the OpenGL tools. ExplorN is freely available from the following ftp site: <ftp://www.galaxy.gmu.edu/pub/software/>.

ExplorN supports scatterplot matrices, parallel coordinate plots, icon-enhanced three-dimensional stereoscopic plots, d -dimensional grand tours and partial grand tours (i.e., tours based on a subset of the variables with the remaining variables being held fixed), and saturation brushing all in a high interaction graphics package.

The ExplorN software is intended to demonstrate principles rather than to be an operational tool so that some refinements normally found in operational software are not there. These include history tracking, easy point identification, identification of mixture weights in the grand tour, relabeling of axes during and after a grand tour as well as simultaneous multiple window views.

Although ExplorN also supports conventional scatterplots and scatterplot matrices, one of its outstanding features are parallel coordinate displays and partial grand tours. Since it is easy to see pairwise relationships for adjacent variables in parallel coordinate plots, but less easy for nonadjacent variables, a complete parallel coordinate investigation would require running through all possible permutations. Instead of this, we recommend using the d -dimensional parallel coordinate grand tour that is implemented in ExplorN. An important interactive procedure for finding clusters using parallel coordinate plots is via the brush-tour.

CrystalVision is a recently developed successor of ExplorN, freely accessible at <ftp://www.galaxy.gmu.edu/pub/software/>. Its main advantage over the older package is that it is available for PCs. Similar to ExplorN, CrystalVision's (see Fig. 10.3) main focus is on parallel coordinate plots, scatterplots, and grand tour animations. Examples of its use, e.g., its EDA techniques applied to scanner data provided by the U.S. Bureau of Labor Statistics (BLS), can be found in Wegman and Dorfman (2003).

Data Viewer, XGobi, and GGobi

In this section we present a series of software developments that was initiated in the mid 1980's by Andreas Buja, Deborah F. Swayne, and Dianne Cook at

the University of Washington, Bellcore, AT&T Bell Labs, and ISU. Other main collaborators that contributed to the development of these software tools that should be mentioned here are Catherine Hurley, John A. McDonald, and Duncan Temple Lang.

The Data Viewer (Buja et al., 1986, 1988; Hurley, 1988, 1989; Hurley and Buja, 1990) is a software package originally developed on a Symbolics Lisp Machine that supports object-oriented programming. The Data Viewer is a system for the exploratory analysis of high-dimensional data sets that allows interactive labeling, identification, brushing, and linked windows. Additional features are viewport transformations such as expanding or shrinking of the data and shifting of the data. The Data Viewer supports several types of projections, including simple 3D rotations, correlation tour (Buja et al., 1988), and grand tour.

Many of the design and layout concepts of the Data Viewer as well as parts of its functionality provided the basic ideas for the follow-up XGobi (see Fig. 10.1), first described in Swayne et al. (1991) and Swayne and Cook (1992). Development on XGobi took place for about a decade; its almost final version is documented in Swayne et al. (1998). XGobi is implemented in the X Windows System, so it runs on any UNIX system, and it runs under Microsoft Windows or the Macintosh operating system if an X emulator is used. XGobi can be freely downloaded from <http://www.research.att.com/areas/stat/xgobi/>.

XGobi is a data visualization system with interactive and dynamic methods for the manipulation of views of data. It offers 2D displays of projections of points and lines in high-dimensional spaces, as well as parallel coordinate plots. Projection tools include dotplots and ASH of single variables, scatterplots of pairs of variables, 3D data rotations, and grand tours. Views of the data can be panned and zoomed. Points can be labeled and brushed with glyphs and colors. Lines can be edited and colored. Several XGobi processes can be run simultaneously and linked for labeling, brushing, and sharing of projections. Missing data are accommodated and their patterns can be examined; multiple imputations can be given to XGobi for rapid visual diagnostics (Swayne and Buja, 1998). XGobi can be cloned, i.e., an identical new XGobi process with exactly the same data and all brushing information can be invoked.

Rotating plots are nowadays implemented in most statistical packages, but the implementation in XGobi goes beyond most of the others. In addition to the standard grand tour, XGobi supports the projection pursuit guided tour. More details on projection pursuit indices available in XGobi can be found in Cook (1993, 1995). Additional index functions that result in speed improvements of the calculations have been presented in Klinke and Cook (1997).

GGobi (Swayne et al., 2003) is a direct descendant of XGobi, but it has been thoroughly redesigned. GGobi (see Fig. 10.2) can be freely downloaded from <http://www.ggobi.org/>.

At first glance, GGobi looks quite unlike XGobi because GGobi uses a newer graphical toolkit called GTK+ (<http://www.gtk.org>), with a more contemporary look and feel and a larger set of user interface components. Through the use of

GTK+, GGobi can be used directly on Microsoft Windows, without any emulator. In addition, GGobi can be used on any UNIX and Linux system.

In contrast to XGobi, the plot window in GGobi has been separated from the control panel. In XGobi, there is in general a single plot per process; to look at multiple views of the same data, we have to launch multiple XGobi processes. In contrast, a single GGobi session can support multiple plots of various types: scatterplots, parallel coordinate plots, scatterplot matrices, and time series plots have been implemented so far. Other changes in GGobi's appearance and repertoire of tools (when compared to XGobi) include an interactive color lookup table manager, the ability to add variables "on the fly", and a new interface for view scaling (panning and zooming). At this point, some of the advanced grand tour and projection pursuit guided tour features from XGobi have not been fully reimplemented in GGobi (but hopefully will be available in the near future).

Other Graphical Software

10.4.4

While the previous sections summarize software that focuses on interactive and dynamic graphics, there exist several statistical languages that provide a tight integration of interactive graphics and numerical computations. Examples for such languages are S/S-PLUS (Becker et al., 1988a; Becker, 1994; Chan, 1997), R (Ihaka and Gentleman, 1996), and XploRe (Härdle et al., 1995). Other examples of software that link interactive graphics, computation, and spread sheets, often through the Web, are the Data Representation System (DRS) by Inoue et al. (2002), DAVIS by Huh and Song (2002), KyPlot by Yoshioka (2002), and the XploRe Quantlet Client/Server (XQC/XQS) architecture (Kleinow and Lehmann, 2002).

Interactive 3D Graphics

10.5

A natural extension of 2D interactive and dynamic graphics is the use of anaglyphs and stereoscopic displays on a computer screen and eventually the use of VR environments to obtain a 3D representation of statistical data and linked objects from geography or medicine.

Anaglyphs

10.5.1

A German teacher, Wilhelm Rollmann, initially described the effect of stereoscopic graphics drawn in red and green colors that are looked at with the naked eye (Rollmann, 1853a), i.e., what is now called free-viewing stereoscopic images. Later the same year, Rollmann (1853b) describes the effect of looking at such colored pictures using filter glasses of corresponding complementary colors. As a reminder, red and green are complementary colors in the conventional color model whereas red and cyan are complementary colors in the RGB color model used for most computer monitors. Eventually, the work by Wilhelm Rollmann has been judged

by Vuibert (1912) and Rösch (1954) as the birth of anaglyphs. The mathematics underlying anaglyphs and stereoscopic displays can be found in Hodges (1992) and Wegman and Carr (1993) for example.

Stereoscopic displays and anaglyphs have been used within statistics by Daniel B. Carr, Richard J. Littlefield, and Wesley L. Nicholson (Carr et al., 1983, 1986; Carr and Littlefield, 1983; Carr and Nicholson, 1985). In particular anaglyphs can be considered an important means to represent three-dimensional pictures on flat surfaces. They have been used in a variety of sciences but found only little use in statistics. One of the first implementations of red-green anaglyphs was the “real-time rotation of three-dimensional scatterplots” in the Mason Hypergraphics software package, described in Bolorfroush and Wegman (1988), page 125. Independently from the work on anaglyphs conducted in the U.S., interactive statistical anaglyph programs also were developed by Franz Hering, Jürgen Symanzik, and Stephan von der Weydt at the Fachbereich Statistik, University of Dortmund (Hering and von der Weydt, 1989; Hering and Symanzik, 1992; Symanzik, 1992, 1993a, 1993b; Hering, 1994).

Wegman and DePriest (1986) is one of the rare sources in statistics where anaglyphs are used in the papers of Banchoff (1986), Carr et al. (1986), and Gabriel and Odoroff (1986). Moreover, Wegman and DePriest (1986) seems to be the first *statistical* reference where colored (red-green) anaglyphs have been published in print.

10.5.2 Virtual Reality

Many different definitions of the term VR can be found throughout the literature. Cruz-Neira (1993) summarizes several possible definitions of VR, including the following working definition for this chapter: “*Virtual reality refers to immersive, interactive, multi-sensory, viewer-centered, three-dimensional computer generated environments and the combination of technologies required to build these environments.*” A brief chronology of events that influenced the development of VR can be found in Cruz-Neira (1993). A more detailed overview on VR can be found in Pimentel and Teixeira (1995) or Vince (1995) for example.

Carolina Cruz-Neira and her colleagues developed an ambitious visualization environment at the Electronic Visualization Lab (EVL) of the University of Illinois in Chicago, known simply as the CAVE (Cruz-Neira et al., 1992, 1993a, 1993b; Cruz-Neira, 1995; Roy et al., 1995). The abbreviation CAVE stands for *CAVE Audio Visual Experience Automatic Virtual Environment*. Carolina Cruz-Neira moved to ISU in 1995 where she was involved in the development of a second, larger CAVE-like environment known as the C2. The CAVE, C2, and several other of its successors belong to immersive projection technology (IPT) systems where the user is visually immersed within the virtual environment.

The use of ISU’s C2 for statistical visualization is based on the framework of three-dimensional projections of p -dimensional data, using as a basis the methods developed and available in XGobi. The implementation of some of the basic XGobi features in the C2 resulted in VRGobi (see Fig. 10.5). The main difference between

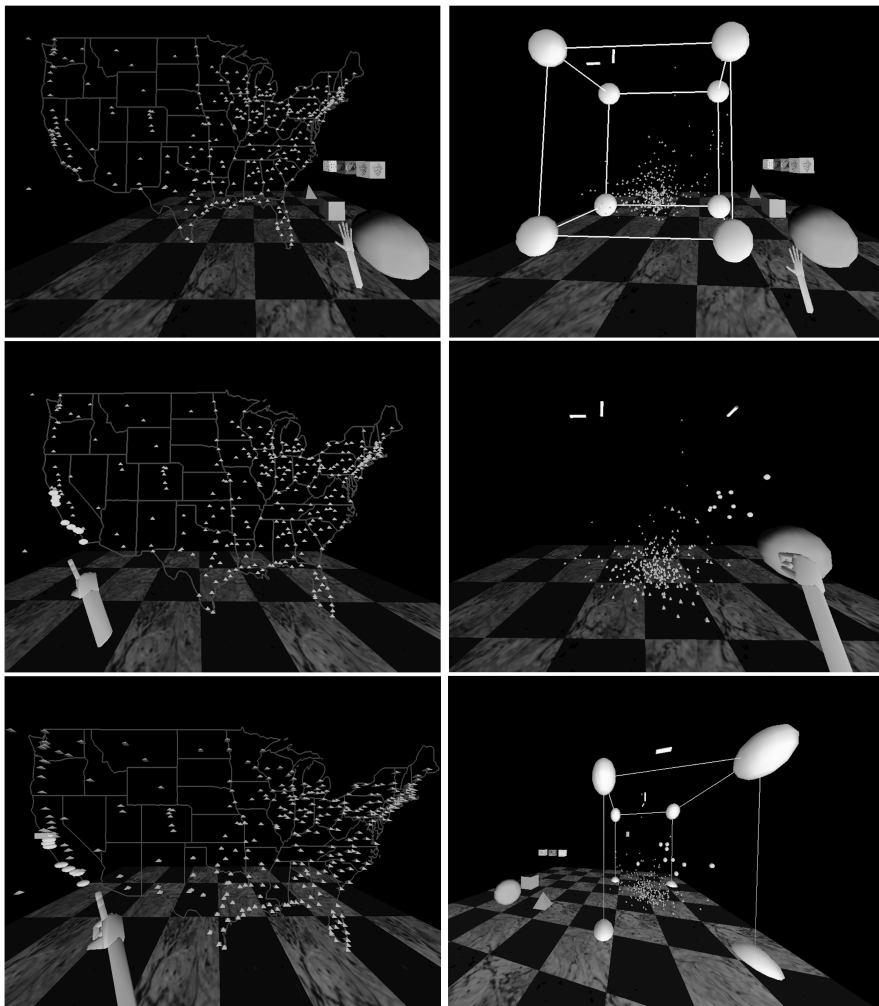


Figure 10.5. Screenshots of the “Places” data in VRGobi, previously published in Symanzik et al. (1996a). A map view (*left*) and a three-dimensional point cloud displaying HousingCost, Climate, and Education are shown (*right*). The control panel, glyph types, and the boundary box that delimits the plot area are visible (*top row*). Cities with nice Climate and high HousingCost have been brushed and happen to fall into California (*middle row*). Among the brushed points is one city (San Francisco) with an outstanding value for Education (*bottom row*). When running VRGobi in the C2 (instead of producing screenshots from one of the control monitors), the rendered arm may be replaced by a human user who is possibly wearing a data glove

XGobi and VRGobi is that the XGobi user interface is rather like a desktop with pages of paper whereas VRGobi is more like having the whole room at the user’s disposal for the data analysis.

VRGobi and the statistical visualization in the C2 have been extensively explored and documented in the literature (Symanzik et al., 1996a, 1997; Cook et al., 1997a, 1998; Nelson et al., 1998, 1999; Cook, 2001). Main developers of VRGobi, over time, were Dianne Cook and Carolina Cruz-Neira, with major contributions by Brad Kohlmeyer, Uli Lechner, Nicholas Lewin, Laura Nelson, and Jürgen Symanzik. Additional information on VRGobi can be found at <http://www.las.iastate.edu/news/Cook0219.html>.

The initial implementation of VRGobi contains a three-dimensional grand tour. Taking arbitrary three-dimensional projections can expose features of the data not visible in one-dimensional or two-dimensional marginal plots.

One of the most difficult developments for VRGobi was the user interface (and not the statistical display components). While it is relatively simple to create popup menus that allow to select colors and symbols for brushing in a desktop environment, designing an appealing and operational three-dimensional interface for the C2 was a real challenge. Eventually, four main components make up VRGobi: the viewing box, the three-dimensional control panel, the variable spheres (similar to the variable circles used in XGobi), and possibly a map view.

A three-dimensional map view, if used, allows the user to explore data in its spatial context within VRGobi, similar to the ArcView/XGobi link (Cook et al., 1996, 1997b) for the desktop.

IPT environments are remarkably different from display devices that are commonly available for data analysis. They extend beyond the small gain of one more dimension of viewing space, to being a completely defined “real” world space. In VRGobi, the temptation is to grab the objects or climb a mountain in the map view and to step aside when a point approaches our face during the grand tour. The objects surround the viewer and it is possible to walk through the data.

In Nelson (1998, 1999), experiments have been conducted on structure detection, visualization, and ease of interaction. Because only 15 human subjects participated in these experiments, it could not be expected that statistically significant results were obtained. However, these experiments showed that there was a clear trend that the test subjects performed considerably better on visualization tasks in the C2 than with XGobi on the workstation display. In contrast, interaction tasks such as brushing provided better results for the workstation. However, subjects with some limited VR experiences already performed considerably better on the interaction tasks in the C2 than subjects with no prior VR experience, suggesting that there is some learning needed to effectively use the VR hardware.

The high cost factor of the CAVE, C2, and similar IPT environments motivated the development of the PC-based MiniCAVE environment. The MiniCAVE is an immersive stereoscopic projection-based VR environment developed at GMU. It is oriented toward group interactions. As such, it is particularly suited to collaborative efforts in scientific visualization, data analysis, and VDM.

Initially researchers began with a 333 megahertz Pentium II machine running Windows NT. The SGI-based VR applications that make use of the OpenGL standard could be ported relatively easily to a PC environment. Using the Windows NT drivers, it was also possible to integrate the Crystal Eyes shutter glasses into

the PC environment. The development of the MiniCAVE, now patented (Patent No. 6,448,965 “Voice-Controlled Immersive Virtual Reality System”) to GMU, has been documented in Wegman et al. (1999) and Wegman and Symanzik (2002).

The one-wall MiniCAVE with speech recognition has been implemented on a dual 450 megahertz Pentium III machine at GMU. In addition, a polarized light LCD projector with both front and rear projection is used. Versions of ExplorN and CrystalVision have been ported to the MiniCAVE environment.

In addition to the work on VR-based data visualization conducted at ISU and GMU, independent work also has been conducted elsewhere, e.g., at Georgia Tech and the Delft Technical University, The Netherlands, resulting in the Virtual Data Visualizer (van Teylingen et al., 1997), and at the University of South Carolina, using the Virtual Reality Modeling Language (VRML) for VR applications on the World Wide Web (Rossini and West, 1998). Böhlen et al. (2003) describe 3DVDM, a 3D VDM system, that is aimed at the visual exploration of large data bases. More details are available at <http://www.cs.auc.dk/3DVDM>.

Cook (2001) lists three fields, “*environmental studies, especially data having a spatial component; shape statistics; and manufacturing quality control*”, that would benefit most from VR and other IPT environments. Certainly, recent experimental desktop links of VR and visualization software with spatial statistical applications such as the links between ViRGIS and RA₃DIO with XGobi (Symanzik et al., 1998b; Schneider et al., 2000) would benefit considerably when being conducted in an IPT environment. In addition to the fields in Cook (2001), we think that medical, genetic, and biological statistical data would also considerably benefit when being explored in an IPT environment.

Applications in Geography, Medicine, and Environmental Sciences

10.6

Geographic Brushing and Exploratory Spatial Data Analysis

10.6.1

Linking statistical plots with geography for analyzing spatially referenced data has been discussed widely in recent years. Monmonier (1988, 1989) describe a conceptual framework for geographical representations in statistical graphics and introduce the term geographic brushing in reference to interacting with the map view of geographically referenced data. But geographic brushing does not only mean pure interaction with the map. In addition, this term has a much broader meaning, e.g., finding neighboring points and spatial structure in a geographic setting.

The idea to apply interactive and dynamic graphics for EDA in a spatial (geographic) context resulted in the term exploratory spatial data analysis (ESDA). However, ESDA is more than just EDA applied to spatial data. Specialized ESDA

methods have been developed that take the special nature of spatial data explicitly into account. ESDA is discussed in more details in Anselin (1998, 1999) and Fotheringham et al. (2000), Chap. 4. Edsall (2003) provides examples for the use of dynamic and interactive parallel coordinate plots for the exploration of large spatial and spatio-temporal data bases.

Many software solutions have been developed that link geographic displays with interactive statistical software packages. In McDonald and Willis (1987), a grand tour is linked to an image to assess the clustering of landscape types in the band space of a LandSat image taken over Manaus, Brazil. In Carr et al. (1987) and Monmonier (1989), a scatterplot matrix is linked to a map view. In REGARD map views are linked with histograms and scatterplots and, moreover, diagnostic plots for assessing spatial dependence are also available. Another exploratory system that links histograms and scatterplots with latitude and longitude (and depth) coordinates is discussed in MacDougall (1992). In Carr et al. (1992), (bivariate) ray-glyph maps have been linked with scatterplots. Nagel (1994) discusses the interactive analysis of spatial data, mostly environmental and disease data, under ISP. Klein and Moreira (1994) report on an interface between the image program MTID and XGobi, used for the exploratory analysis of agricultural images. DiBiase et al. (1994) provide an overview on existing multivariate (statistical) displays for geographic data. Other developments are the cartographic data visualizer, *cdv* (Dykes, 1996), where a variety of plots are linked with geography, the Space-Time-Attribute Creature/Movie, STAC/M (Openshaw and Perrée, 1996), that searches for patterns in Geographic Information System (GIS) data bases under the control of a Genetic Algorithm, and an exploratory spatial analysis system in XLisp-Stat (Brunsdon and Charlton, 1996).

In combination with the GIS ArcView, XGobi and XploRe also have been used to detect structure and abnormalities in geographically referenced data sets such as satellite imagery, forest health monitoring, and precipitation data (Cook et al., 1996, 1997b; Symanzik et al., 1996b, 1998a, 2000a) (see Fig. 10.1). In addition to the ArcView/XGobi/XploRe environment, there are several other examples where GIS's and (graphical) statistical packages have been linked. Williams et al. (1990) demonstrate how S and the GRASS GIS can be jointly used for archaeological site classification and analysis. Scott (1994) links STATA with ArcView. The spatial data analysis software SpaceStat has been linked with ARC/INFO (Anselin et al., 1993) and with ArcView (Anselin and Bao, 1996, 1997). In Haining et al. (1996), the designing of a software system for interactive exploration of spatial data by linking to ARC/INFO has been discussed, and in Zhang and Griffith (1997), a spatial statistical analysis module implemented in ArcView using Avenue has been discussed. MathSoft (1996) describes the S+GISLink, a bidirectional link between ARC/INFO and S-PLUS, and Bao (1997) describes the S+Grassland link between S-PLUS and the Grassland GIS. Finally, a comparison of the operational issues of the SpaceStat/ArcView link and the S+Grassland link has been given in Bao and Anselin (1997).

Interactive Micromaps

Over the last decade, researchers have developed many improvements to make statistical graphics more accessible to the general public. These improvements include making statistical summaries more visual and providing more information at a time. Research in this area involved converting statistical tables into plots (Carr, 1994; Carr and Nusser, 1995), new ways of displaying geographically referenced data (Carr et al., 1992), and, in particular, the development of linked micromap (LM) plots (see Fig. 10.6), often simply called micromaps (Cherry and Phelps, 1996; Carr et al. 1998, 2000a). LM plots were first presented in a poster session sponsored by the ASA Section on Statistical Graphics at the 1996 Joint Statistical Meetings in Chicago (“Presentation of Data in Linked Attribute and Geographic Space” by Anthony R. Olsen, Daniel B. Carr, Jean-Yves P. Courbois, and Suzanne Pierson). More details on the history of LM plots and their connection to other research can be found in these early references on micromaps. Recent references on LM plots (Carr et al., 2000b; Carr, 2001) focus on their use for communicating summary data from health and environmental studies. Sample S-PLUS code, data files, and resulting plots from Daniel B. Carr’s early micromap articles can be accessed at <ftp://galaxy.gmu.edu/pub/dcarr/newsletter/micromap/>.

Linked micromap plots provide a new statistical paradigm for the viewing geographically referenced statistical summaries in the corresponding spatial context. The main idea behind LM plots is to focus the viewer’s attention on the statistical information presented in a graphical display. Multiple small maps are used to provide the appropriate geographic reference for the statistical data.

Initially, LM plots were only static representations on paper. The next stage of LM plots was aimed at interactive displays on the Web. Eventually, generalized maps for all states in the U.S. and several counties were automatically created for use on the U.S. Environmental Protection Agency (EPA) Cumulative Exposure Project (CEP) Web site (Symanzik et al., 2000b). Most current applications of interactive LM plots on the Web make use of these generalized maps.

The idea of using micromaps on the Web was first considered for the EPA CEP Web site (formerly accessible at <http://www.epa.gov/CumulativeExposure/>). Initially, the EPA wanted to provide fast and convenient Web-based access to its hazardous air pollutant (HAP) data for 1990. Unfortunately, no part of the interactive CEP Web site was ever published due to concerns that the 1990 data was outdated at the intended release date in 1998. Only a static version of the CEP Web site without tables and micromaps was accessible. More details on the work related to the planned interactive CEP Web site can be found in Symanzik (1999b, 1999c, 2000b).

The U.S. Department of Agriculture (USDA) – National Agricultural Statistics Service (NASS) Research and Development Division released a Web site (<http://www.nass.usda.gov/research/sumpant.htm>) in September 1999 that uses interactive micromaps to display data from the 1997 Census of Agriculture. While the end user who accesses this Web site gets the impression of fully interactive graphics, this is not the case. The 10 micromaps (5 crops \times 2 arrangements) plus

Education and Crime Index of Selected Cities (by State)

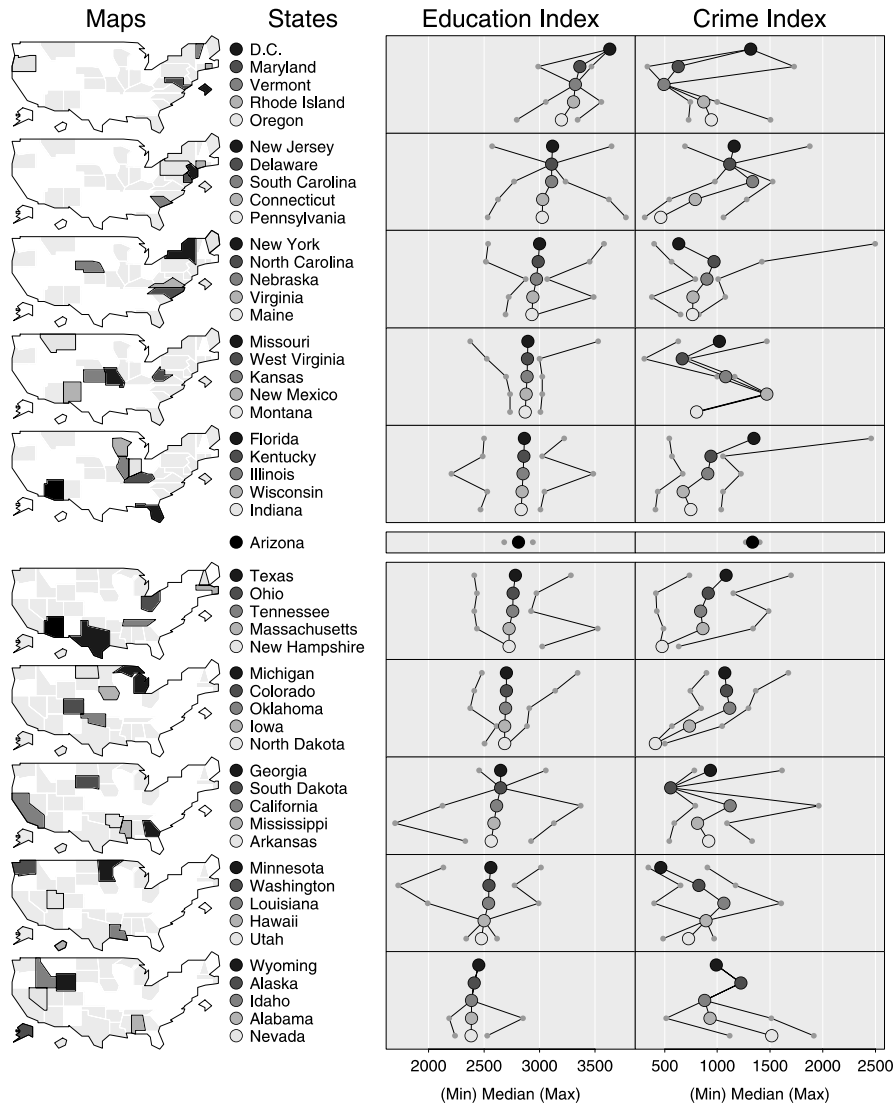


Figure 10.6. Linked micromap plot of the “Places” data, adapted from Daniel B. Carr’s sample S-PLUS code. The variables Education and Crime have been summarized at the state level for this figure. For each of the 50 states (plus Washington, D.C.), the minimum, median, and maximum of the Education and Crime indexes have been obtained for the cities that geographically belong to a state. It should be noted that for several of the states data exist for only one city. The map and statistical displays have been sorted with respect to decreasing median Education Index. The zig-zag curve of the related median Crime Index is an indicator of little correlation between these two variables. Numerically, the ecological correlation between median Education Index and median Crime Index is almost equal to zero

one overview micromap were precalculated in S-PLUS and were stored as jpg images. It is not possible to create any new micromap display “on the fly” on this Web site.

The National Cancer Institute (NCI) released a Web site in April 2003 that provides interactive access to its cancer data via micromaps. This Web site is Java-based and creates micromaps “on the fly”. Wang et al. (2002) and Carr et al. (2002) provide more details on the design of the NCI Web site that is accessible at <http://www.statecancerprofiles.cancer.gov/micromaps>.

nViZn (read envision) is a JAVA-based software development kit (SDK), currently developed and distributed by SPSS (<http://spss.com/nvizn/>). It is the follow-up to the Graphics Production Library (GPL), described in Carr et al. (1996), developed within the BLS. nViZn (Wilkinson et al., 2000) is based on a formal grammar for the specification of statistical graphics (Wilkinson, 1999), see also Chap. II.11. In addition to capabilities already present in the original GPL, nViZn has many additional features. Most useful for the display of data in a geographic context are the capabilities that enable a programmer to create interactive tables and linked micromaps in nViZn. Experiences with nViZn, its advantages and current problems, and its capabilities for the display of Federal data via LM plots are described in more detail in Jones and Symanzik (2001), Symanzik and Jones (2001), and Symanzik et al. (2002a).

Micromap implementations that allow the user to create new LM plots “on the fly” often provide features to switch from one geographic region or subregion to another, choose among several variables, resort the data increasingly or decreasingly according to different statistics (such as mean, median, minimum, or maximum of the data values in the underlying geographic region), and display different graphical summaries of the data (e.g., dotplots, boxplots, confidence intervals, or even time series). So, in an interactive environment, a user might want to create a LM plot of Education and Arts (sorted by increasing maximum Education) after having studied the LM plot in Fig. 10.6 – and then immediately resort the display by decreasing maximum Arts.

Conditioned Choropleth Maps

10.6.3

Conditioned choropleth maps (CCmaps), described in Carr et al. (2000b, 2002), focus on spatial displays that involve one dependent variable and two independent variables. CCmaps promote interactive hypothesis generation, common for epidemiological and environmental applications. In fact, applications from the National Center for Health Statistics (NCHS) and the EPA motivated the development of CCmaps. CCmaps are written in Java and can be freely obtained from <http://www.galaxy.gmu.edu/~dcarr/ccmaps>.

The main interactive component of CCmaps are partitioning sliders that allow the user to dynamically partition the study units into a 3×3 layout of maps. The sliders allow a user to create, examine, and contrast subsets for the purpose of generating hypotheses about patterns in spatially referenced data. For example, in a medical application one of the sliders might control the age intervals and

the second slider might control the years of active smoking in a study on cancer mortality rates across the U.S. The resulting 9 maps will allow an analyst to develop hypotheses on spatial patterns within panels or among panels. Additional features of this CCmaps implementation are dynamic quantile-quantile (QQ) plots and pan and zoom widgets to allow closer inspection of data at the U.S. county level.

10.7 Outlook

10.7.1 Limitations of Graphics

Wegman (1995) discusses aspects of data set size, computational feasibility, and in particular limits of visualization for “large” (about 10^8 bytes) and “huge” (about 10^{10} bytes) data sets, where “large” and “huge” are terms introduced in Huber’s taxonomy of large data sets (Huber, 1992, 1994). As pointed out in Wegman (1995), even in the most wildly optimistic scenario, i.e., an angular resolution of 4.38 minutes of arc as suggested in Maar (1982), immersion, and a 4:5 aspect-ratio, the human eye would only be able to distinguish $17,284 \times 13,828 = 2.39 \times 10^8$ pixels. Using single-pixel coding, it seems to be impossible to visualize “large” to “huge” data sets.

Huber (1994) initially suggests to prepare “medium” (about 10^6 bytes) derived data sets that are easier to visualize and grasp as a whole and can still be worked with established techniques of high interaction graphics. Unfortunately, as Wegman (1995) further describes, common ways of parsing data sets down, e.g., clustering, discriminant analysis, and principal components, are computationally complex (often of a magnitude of $O(n^{3/2})$ or even $O(n^2)$) and therefore are not valid alternatives. It seems that simple random thinning is the only methodology of choice, but this may have the side effect of missing some of the tail structure an analyst may actually be looking for.

Possible solutions are the use of 3D VR techniques that may display up to 10^{10} voxels (Wegman, 1995), further advances in selection sequences, or new strategies to increase visual scalability i.e., the capability of visualization tools to effectively display large data sets (Eick and Karr, 2002). However, the conclusion in Wegman (1995) that “*visualization of data sets say of size 10^6 or more is clearly a wide open field*” is still valid today.

10.7.2 Future Developments

Historically, one of the problems with interactive and dynamic statistical graphics was to publish visible results. The ASA Statistical Graphics Section Video Lending Library was one attempt to capture at least some snapshots of software and applications of interactive and dynamic statistical graphics and preserve them for the future. Publishing a sequence of screenshots in a written paper clearly has not

the same effect as watching the full interaction and being able to manipulate the graphics.

Recently, many books have been published with accompanying CDs and DVDs and many conference proceedings have been published on CD. Due to this trend, it is now possible to immediately publish a movie accompanying a written paper or integrate interactive graphics within a paper. Examples are Wojciechowski and Scott (2000) and Symanzik et al. (2002b) where the papers are accompanied by several movie segments. It is expected that more and more future publications on interactive and dynamic graphics will be accompanied by an interactive application or by movies.

To be useful for the future, interactive and dynamic graphics have to adapt to challenges posed by “large” and “huge” (in terms of Huber’s taxonomy) data sets as outlined in Sect. 10.7.1. Examples for such data sets are data from earth or planetary observation systems, real-time stream data (such as from surveillance systems), or any other type of massive data sets. Wegman (2000) provides a futuristic vision, indicating that major advancements can be expected in DM, visualization, and quantization methods.

For smaller data sets (from “tiny” to “medium” in terms of Huber’s taxonomy) we can expect to see progress in new user paradigms that will allow to interact with the data through voice or gestures as well as multiple users to manipulate the visible view simultaneously. It can also be expected that more graphical software will make use of the Web and mobile data transmission and reception techniques. The same software may therefore be available for a variety of hardware platforms with screens as small as a clock or cell phone or as big as a 3D IMAX theatre.

As pointed out in Carr et al. (2002), “*there are many barriers to acceptance of new methodology by federal agencies.*” This can be easily extended towards other users of newly developed interactive and dynamic graphical software. Clearly, just promoting a new idea or graphical software product is not enough in many cases. It is likely that more usability tests of new graphical software products as well as comparative reviews of old and new tools will be conducted in the future.

References

- Andrews, D.F. (1972). Plots of High-Dimensional Data. *Biometrics*, 28:125–136.
- Anselin, L. (1998). Exploratory Spatial Data Analysis in a Geocomputational Environment. In Longley, P.A., Brooks, S.M., McDonnell, R., and Macmillan, B., editors, *Geocomputation – A Primer*, pages 77–94. Wiley, Chichester.
- Anselin, L. (1999). Interactive Techniques and Exploratory Spatial Data Analysis. In Longley, P.A., Goodchild, M.F., Maguire, D.J., and Rhind, D.W., editors, *Geographical Information Systems, Volume 1: Principles and Technical Issues (Second Edition)*, pages 253–266. Wiley, New York, NY.
- Anselin, L. and Bao, S. (1996). Exploratory Spatial Data Analysis Linking SpaceStat and ArcView. Technical Report 9618, West Virginia University, Morgantown, WV.

- Anselin, L. and Bao, S. (1997). Exploratory Spatial Data Analysis Linking SpaceStat and ArcView. In Fischer, M.M. and Getis, A., editors, *Recent Developments in Spatial Analysis*, pages 35–59. Springer, Berlin.
- Anselin, L., Dodson, R.F., and Hudak, S. (1993). Linking GIS and Spatial Data Analysis in Practice. *Geographical Systems*, 1(1):3–23.
- Asimov, D. (1985). The Grand Tour: A Tool for Viewing Multidimensional Data. *SIAM Journal on Scientific and Statistical Computing*, 6(1):128–143.
- Banchoff, T.F. (1986). Visualizing Two-Dimensional Phenomena in Four-Dimensional Space: A Computer Graphics Approach. In Wegman, E.J. and DePriest, D.J., editors, *Statistical Image Processing and Graphics*, pages 187–202. Marcel Dekker, New York, NY.
- Bao, S. (1997). *User's Reference for the S+Grassland Link*. Mathsoft, Inc., Seattle, WA.
- Bao, S. and Anselin, L. (1997). Linking Spatial Statistics with GIS: Operational Issues in the SpaceStat-ArcView Link and the S+Grassland Link. In *1997 Proceedings of the Section on Statistical Graphics*, pages 61–66, Alexandria, VA. American Statistical Association.
- Becker, R.A. (1994). A Brief History of S. In Dirschedl, P. and Ostermann, R., editors, *Computational Statistics*, pages 81–110. Physica-Verlag, Heidelberg.
- Becker, R.A., Chambers, J.M., and Wilks, A.R. (1988a). *The New S Language – A Programming Environment for Data Analysis and Graphics*. Wadsworth and Brooks/Cole, Pacific Grove, CA.
- Becker, R.A. and Cleveland, W.S. (1988). Brushing Scatterplots. In Cleveland, W.S. and McGill, M.E., editors, *Dynamic Graphics for Statistics*, pages 201–224. Wadsworth & Brooks/Cole, Belmont, CA.
- Becker, R.A., Cleveland, W.S., and Weil, G. (1988b). The Use of Brushing and Rotation for Data Analysis. In Cleveland, W.S. and McGill, M.E., editors, *Dynamic Graphics for Statistics*, pages 247–275. Wadsworth & Brooks/Cole, Belmont, CA.
- Blasius, J. and Greenacre, M., editors (1998). *Visualization of Categorical Data*. Academic Press, San Diego, CA.
- Böhlen, M., Bukauskas, L., Eriksen, P.S., Lauritzen, S.L., Mazeika, A., Musaeus, P., and Mylov, P. (2003). 3D Visual Data Mining – Goals and Experiences. *Computational Statistics & Data Analysis: Special Issue on Data Visualization*, 43(4):445–469.
- Bolorforoush, M. and Wegman, E.J. (1988). On Some Graphical Representations of Multivariate Data. In Wegman, E.J., Gantz, D.T., and Miller, J.J., editors, *Proceedings of the 20th Symposium on the Interface between Computing Science and Statistics*, pages 121–126. American Statistical Association, Alexandria, VA.
- Boyer, R. and Savageau, D. (1981). *Places Rated Almanac*. Rand McNally, Chicago, IL.
- Brunsdon, C. and Charlton, M. (1996). Developing an Exploratory Spatial Analysis System in XLisp-Stat. In Parker, D., editor, *Innovations in GIS 3*, pages 135–145. Taylor & Francis, London, U.K.
- Buja, A. and Asimov, D. (1986a). Grand Tour Methods: An Outline. *Computing Science and Statistics*, 17:63–67.

- Buja, A. and Asimov, D. (1986b). Grand Tour Methods: An Outline. In Allen, D.M., editor, *Proceedings of the 17th Symposium on the Interface between Computer Science and Statistics*, Lexington, KY, pages 63–67. Elsevier.
- Buja, A., Asimov, D., Hurley, C., and McDonald, J.A. (1988). Elements of a Viewing Pipeline for Data Analysis. In Cleveland, W.S. and McGill, M.E., editors, *Dynamic Graphics for Statistics*, pages 277–308. Wadsworth & Brooks/Cole, Belmont, CA.
- Buja, A., Cook, D., and Swayne, D.F. (1996). Interactive High-Dimensional Data Visualization. *Journal of Computational and Graphical Statistics*, 5(1):78–99.
- Buja, A., Hurley, C., and McDonald, J.A. (1986). A Data Viewer for Multivariate Data. In Boardman, T.J. and Stefanski, I.M., editors, *Proceedings of the 18th Symposium on the Interface between Computer Science and Statistics*, Fort Collins, CO, pages 171–174. American Statistical Association, Washington, D.C.
- Buja, A., McDonald, J.A., Michalak, J., and Stuetzle, W. (1991). Interactive Data Visualization Using Focusing and Linking. In Nielson, G.M. and Rosenblum, L.J., editors, *Proceedings of Visualization '91*, Los Alamitos, CA, pages 156–163. IEEE Computer Society Press.
- Buja, A. and Tukey, P.A., editors (1991). *Computing and Graphics in Statistics*. Springer, New York, NY.
- Carr, D.B. (1991). Looking at Large Data Sets Using Binned Data Plots. In Buja, A. and Tukey, P.A., editors, *Computing and Graphics in Statistics*, pages 7–39. Springer, New York, NY.
- Carr, D.B. (1994). Converting Tables to Plots. Technical Report 101, Center for Computational Statistics, George Mason University, Fairfax, VA.
- Carr, D.B. (2001). Designing Linked Micromap Plots for States with Many Counties. *Statistics in Medicine*, 20(9–10):1331–1339.
- Carr, D.B., Chen, J., Bell, B.S., Pickle, L., and Zhang, Y. (2002). Interactive Linked Micromap Plots and Dynamically Conditioned Choropleth Maps. In *dg.02002 Proceedings*. Digital Government Research Center (DGRC). http://www.dgrc.org/conferences/2002_proceedings.jsp.
- Carr, D.B. and Littlefield, R.J. (1983). Color Anaglyph Stereo Scatterplots – Construction Details. In Gentle, J.E., editor, *Proceedings of the 15th Symposium on the Interface between Computer Science and Statistics*, pages 295–299, New York, NY. North-Holland Publishing Company.
- Carr, D.B., Littlefield, R.J., and Nicholson, W.L. (1983). Color Anaglyph Stereo Scatterplots – Construction and Application. In *1983 Proceedings of the Section on Statistical Computing*, pages 255–257, Alexandria, VA. American Statistical Association.
- Carr, D.B., Littlefield, R.J., Nicholson, W.L., and Littlefield, J.S. (1987). Scatterplot Matrix Techniques for Large N. *Journal of the American Statistical Association*, 82(398):424–436.
- Carr, D.B. and Nicholson, W.L. (1985). Evaluation of Graphical Techniques for Data in Dimensions 3 to 5: Scatterplot Matrix, Glyph, and Stereo Examples. In *1985 Proceedings of the Section on Statistical Computing*, pages 229–235, Alexandria, VA. American Statistical Association.

- Carr, D.B. and Nicholson, W.L. (1988). EXPLOR4: A Program for Exploring Four-Dimensional Data Using Stereo-Ray Glyphs, Dimensional Constraints, Rotation, and Masking. In Cleveland, W.S. and McGill, M.E., editors, *Dynamic Graphics for Statistics*, pages 309–329. Wadsworth & Brooks/Cole, Belmont, CA.
- Carr, D.B., Nicholson, W.L., Littlefield, R.J., and Hall, D.L. (1986). Interactive Color Display Methods for Multivariate Data. In Wegman, E.J. and DePriest, D.J., editors, *Statistical Image Processing and Graphics*, pages 215–250. Marcel Dekker, New York, NY.
- Carr, D.B. and Nusser, S.M. (1995). Converting Tables to Plots: A Challenge from Iowa State. *Statistical Computing and Statistical Graphics Newsletter*, 6(3):11–18.
- Carr, D.B., Olsen, A.R., Courbois, J.P., Pierson, S.M., and Carr, D.A. (1998). Linked Micromap Plots: Named and Described. *Statistical Computing and Statistical Graphics Newsletter*, 9(1):24–32.
- Carr, D.B., Olsen, A.R., Pierson, S.M., and Courbois, J.P. (2000a). Using Linked Micromap Plots to Characterize Omernik Ecoregions. *Data Mining and Knowledge Discovery*, 4(1):43–67.
- Carr, D.B., Olsen, A.R., and White, D. (1992). Hexagon Mosaic Maps for Displays of Univariate and Bivariate Geographical Data. *Cartography and Geographic Information Systems*, 19(4):228–236, 271.
- Carr, D.B. and Pierson, S.M. (1996). Emphasizing Statistical Summaries and Showing Spatial Context with Micromaps. *Statistical Computing and Statistical Graphics Newsletter*, 7(3):16–23.
- Carr, D.B., Valliant, R., and Rope, D.J. (1996). Plot Interpretation and Information Webs: A Time-Series Example from the Bureau of Labor Statistics. *Statistical Computing and Statistical Graphics Newsletter*, 7(2):19–26.
- Carr, D.B., Wallin, J.F., and Carr, D.A. (2000b). Two New Templates for Epidemiology Applications: Linked Micromap Plots and Conditioned Choropleth Maps. *Statistics in Medicine*, 19(17–18):2521–2538.
- Carr, D.B., Wegman, E.J., and Luo, Q. (1997). ExplorN: Design Considerations Past and Present. Technical Report 137, Center for Computational Statistics, George Mason University, Fairfax, VA.
- Chambers, J.M. (1997). Evolution of the S Language. *Computing Science and Statistics*, 28:331–337.
- Chambers, J.M., Cleveland, W.S., Kleiner, B., and Tukey, P.A. (1983). *Graphical Methods for Data Analysis*. Wadsworth & Brooks/Cole, Pacific Grove, CA.
- Chang, J. (1970). Real-Time Rotation. ASA Statistical Graphics Video Lending Library (contact: dj@research.bell-labs.com).
- Cleveland, W.S. (1985). *The Elements of Graphing Data*. Wadsworth, Monterey, CA.
- Cleveland, W.S. and McGill, M.E., editors (1988). *Dynamic Graphics for Statistics*. Wadsworth & Brooks/Cole, Belmont, CA.
- Cook, D. (1997). Calibrate Your Eyes to Recognize High-Dimensional Shapes from Their Low-Dimensional Projections. *Journal of Statistical Software*, 2(6). <http://www.jstatsoft.org/v02/106/>.
- Cook, D. (2001). Virtual Reality: Real Ponderings. *Chance*, 14(1):47–51.

- Cook, D. and Buja, A. (1997). Manual Controls for High-Dimensional Data Projections. *Journal of Computational and Graphical Statistics*, 6(4):464–480.
- Cook, D., Buja, A., and Cabrera, J. (1993). Projection Pursuit Indexes Based on Orthonormal Function Expansions. *Journal of Computational and Graphical Statistics*, 2(3):225–250.
- Cook, D., Buja, A., Cabrera, J., and Hurley, C. (1995). Grand Tour and Projection Pursuit. *Journal of Computational and Graphical Statistics*, 4(3):155–172.
- Cook, D., Cruz-Neira, C., Kohlmeyer, B.D., Lechner, U., Lewin, N., Nelson, L., Olsen, A.R., Pierson, S.M., and Symanzik, J. (1998). Exploring Environmental Data in a Highly Immersive Virtual Reality Environment. *Environmental Monitoring and Assessment*, 51(1/2):441–450.
- Cook, D., Cruz-Neira, C., Lechner, U., Nelson, L., Olsen, A.R., Pierson, S.M., and Symanzik, J. (1997a). Using Dynamic Statistical Graphics in a Highly Immersive Virtual Reality Environment to Understand Multivariate (Spatial) Data. In *Bulletin of the International Statistical Institute, 51st Session Istanbul 1997, Proceedings Book 2*, pages 31–34.
- Cook, D., Majure, J.J., Symanzik, J., and Cressie, N. (1996). Dynamic Graphics in a GIS: Exploring and Analyzing Multivariate Spatial Data Using Linked Software. *Computational Statistics: Special Issue on Computeraided Analysis of Spatial Data*, 11(4):467–480.
- Cook, D., Symanzik, J., Majure, J.J., and Cressie, N. (1997b). Dynamic Graphics in a GIS: More Examples Using Linked Software. *Computers and Geosciences: Special Issue on Exploratory Cartographic Visualization*, 23(4):371–385. Paper, CD, and <http://www.elsevier.nl/locate/cgvis>.
- Cox, K.C., Eick, S.G., Wills, G.J., and Brachman, R.J. (1997). Visual Data Mining: Recognizing Telephone Calling Fraud. *Data Mining and Knowledge Discovery*, 1:225–231.
- Craig, P., Haslett, J., Unwin, A., and Wills, G. (1989). Moving Statistics – An Extension of “Brushing” for Spatial Data. In Berk, K. and Malone, L., editors, *Proceedings of the 21st Symposium on the Interface between Computing Science and Statistics*, pages 170–174. American Statistical Association, Alexandria, VA.
- Crawford, S.L. and Fall, T.C. (1990). Projection Pursuit Techniques for Visualizing High-Dimensional Data Sets. In Nielson, G.M., Shrivvers, B., and Rosenblum, L.J., editors, *Proceedings of Visualization in Scientific Computing, Los Alamitos, CA*, pages 94–108. IEEE Computer Society Press.
- Cruz-Neira, C. (1993). Virtual Reality Overview. SIGGRAPH '93 Course Notes #23. pp. 1–18.
- Cruz-Neira, C. (1995). *Projection-based Virtual Reality: The CAVE and its Applications to Computational Science*. PhD thesis, University of Illinois at Chicago.
- Cruz-Neira, C., Leigh, J., Papka, M., Barnes, C., Cohen, S.M., Das, S., Engelmann, R., Hudson, R., Roy, T., Siegel, L., Vasilakis, C., DeFanti, T., and Sandin, D.J. (1993a). Scientists in Wonderland: A Report on Visualization Applications in the CAVE Virtual Reality Environment. In *IEEE 1993 Symposium on Research Frontiers in Virtual Reality*, pages 59–66.

- Cruz-Neira, C., Sandin, D.J., and DeFanti, T.A. (1993b). Surround-Screen Projection-Based Virtual Reality: The Design and Implementation of the CAVE. In *ACM SIGGRAPH '93 Proceedings*, pages 135–142, Anaheim, CA.
- Cruz-Neira, C., Sandin, D.J., DeFanti, T.A., Kenyon, R.V., and Hart, J.C. (1992). The CAVE: AudioVisual Experience Automatic Virtual Environment. *Communications of the ACM*, 35(6):64–72.
- DiBiase, D., Reeves, C., MacEachren, A.M., von Wyss, M., Krygier, J.B., Sloan, J.L., and Detweiler, M.C. (1994). Multivariate Display of Geographic Data: Applications in Earth System Science. In MacEachren, A.M. and Taylor, D. R.F., editors, *Visualization in Modern Cartography*, pages 287–312. Pergamon (Elsevier), Oxford, U.K.
- du Toit, S. H.C., Steyn, A. G.W., and Stumpf, R.H. (1986). *Graphical Exploratory Data Analysis*. Springer, New York, NY.
- Dykes, J.A. (1996). Dynamic Maps for Spatial Science: A Unified Approach to Cartographic Visualization. In Parker, D., editor, *Innovations in GIS 3*, pages 177–187. Taylor & Francis, London, U.K.
- Edsall, R.M. (2003). The Parallel Coordinate Plot in Action: Design and Use for Geographic Visualization. *Computational Statistics & Data Analysis: Special Issue on Data Visualization*, 43(4):605–619.
- Eick, S.G. and Karr, A.F. (2002). Visual Scalability. *Journal of Computational and Graphical Statistics*, 11(1):22–43.
- FisherKeller, M.A., Friedman, J.H., and Tukey, J.W. (1974a). PRIM-9: An Interactive Multidimensional Data Display and Analysis System. ASA Statistical Graphics Video Lending Library (contact: dj@research.bell-labs.com).
- FisherKeller, M.A., Friedman, J.H., and Tukey, J.W. (1974b). PRIM-9: An Interactive Multidimensional Data Display and Analysis System. Technical Report SLAC-PUB-1408, Stanford Linear Accelerator Center, Stanford, CA.
- Foley, J.D., van Dam, A., Feiner, S.K., and Hughes, J.F. (1990). *Computer Graphics – Principles and Practice (Second Edition)*. Addison-Wesley, Reading, MA.
- Fotheringham, A.S., Brunson, C., and Charlton, M., editors (2000). *Quantitative Geography: Perspectives on Spatial Data Analysis*. Sage, London.
- Friedman, J.H. (1987). Exploratory Projection Pursuit. *Journal of the American Statistical Association*, 82:249–266.
- Friedman, J.H. (1998). Data Mining and Statistics: What's the Connection? *Computing Science and Statistics*, 29(1):3–9.
- Friedman, J.H. and Tukey, J.W. (1974). A Projection Pursuit Algorithm for Exploratory Data Analysis. *IEEE Transactions on Computing C*, 23:881–889.
- Furnas, G.W. (1988). Dimensionality Constraints on Projection and Section Views of High Dimensional Loci. In Wegman, E.J., Gantz, D.T., and Miller, J.J., editors, *Proceedings of the 20th Symposium on the Interface between Computing Science and Statistics*, pages 99–107. American Statistical Association, Alexandria, VA.
- Furnas, G.W. and Buja, A. (1994). Prosection Views: Dimensional Inference Through Sections and Projections (with Discussion). *Journal of Computational and Graphical Statistics*, 3(4):323–385.

- Gabriel, K.R. and Odoroff, C.L. (1986). Illustrations of Model Diagnosis by Means of Three-Dimensional Biplots. In Wegman, E.J. and DePriest, D.J., editors, *Statistical Image Processing and Graphics*, pages 257–274. Marcel Dekker, New York, NY.
- Haining, R., Ma, J., and Wise, S. (1996). Design of a Software System for Interactive Spatial Statistical Analysis Linked to a GIS. *Computational Statistics: Special Issue on Computeraided Analysis of Spatial Data*, 11(4):449–466.
- Hall, P. (1989). Polynomial Projection Pursuit. *Annals of Statistics*, 17:589–605.
- Härdle, W., Klinke, S., and Turlach, B.A. (1995). *XploRe: An Interactive Statistical Computing Environment*. Springer, New York, NY.
- Hering, F. (1994). Anaglyphs in Statistics – A Tool for Interactive Multivariate Data Analysis. In Faulbaum, F., editor, *SoftStat '93 – Advances in Statistical Software 4*, pages 277–283, Stuttgart, Jena, New York. Gustav Fischer.
- Hering, F. and Symanzik, J. (1992). Anaglyphen 3D – Ein Programm zur interaktiven Anaglyphendarstellung. Forschungsbericht 92/1, Fachbereich Statistik, Universität Dortmund, (In German).
- Hering, F. and von der Weydt, S. (1989). Interaktive Anaglyphendarstellungen als Hilfsmittel zur Analyse mehrdimensionaler Daten. Forschungsbericht 89/7, Fachbereich Statistik, Universität Dortmund, (In German).
- Hodges, L.F. (1992). Tutorial: Time-Multiplexed Stereoscopic Computer Graphics. *IEEE Computer Graphics & Applications*, 12(2):20–30.
- Hofmann, H. (2000). Exploring Categorical Data: Interactive Mosaic Plots. *Metrika*, 51(1):11–26.
- Hofmann, H. (2003). Constructing and Reading Mosaicplots. *Computational Statistics & Data Analysis: Special Issue on Data Visualization*, 43(4):565–580.
- Hofmann, H. and Theus, M. (1998). Selection Sequences in MANET. *Computational Statistics: Special Issue on Strategies for Data Analysis*, 13(1):77–87.
- Huber, P.J. (1985). Projection Pursuit (with Discussion). *Annals of Statistics*, 13:435–525.
- Huber, P.J. (1992). Issues in Computational Data Analysis. In Dodge, Y. and Whitaker, J., editors, *COMPSTAT 1992: Proceedings in Computational Statistics, Volume 2*, pages 3–13, Heidelberg. Physica-Verlag.
- Huber, P.J. (1994). Huge Data Sets. In Dutter, R. and Grossmann, W., editors, *COMPSTAT 1994: Proceedings in Computational Statistics*, pages 3–13, Heidelberg. Physica-Verlag.
- Huh, M.Y. and Song, K. (2002). DAVIS: A Java-Based Data Visualization System. *Computational Statistics*, 17(3):411–423.
- Hummel, J. (1996). Linked Bar Charts: Analysing Categorical Data Graphically. *Computational Statistics*, 11:23–33.
- Hurley, C. (1988). A Demonstration of the Data Viewer. In Wegman, E.J., Gantz, D.T., and Miller, J.J., editors, *Proceedings of the 20th Symposium on the Interface between Computing Science and Statistics*, pages 108–114. American Statistical Association, Alexandria, VA.
- Hurley, C. (1989). *The Data Viewer: A Program for Graphical Data Analysis*. PhD thesis, Statistics Department, University of Washington, Seattle.

- Hurley, C. and Buja, A. (1990). Analyzing High-Dimensional Data with Motion Graphics. *SIAM Journal on Scientific and Statistical Computing*, 11(6):1193–1211.
- Ihaka, R. and Gentleman, R. (1996). R: A Language for Data Analysis and Graphics. *Journal of Computational and Graphical Statistics*, 5(3):299–314.
- Inoue, T., Asahi, Y., Yadohisa, H., and Yamamoto, Y. (2002). A Statistical Data Representation System on the Web. *Computational Statistics*, 17(3):367–378.
- Inselberg, A. (1985). The Plane with Parallel Coordinates. *The Visual Computer*, 1:69–91.
- Inselberg, A. (1998). Visual Data Mining with Parallel Coordinates. *Computational Statistics: Special Issue on Strategies for Data Analysis*, 13(1):47–63.
- Jones, L. and Symanzik, J. (2001). Statistical Visualization of Environmental Data on the Web using nViZn. *Computing Science and Statistics*, 33. (CD).
- Jones, M.C. and Sibson, R. (1987). What is Projection Pursuit? (with Discussion). *Journal of the Royal Statistical Society, Series A*, 150:1–36.
- Klein, R. and Moreira, R.I. (1994). Exploratory Analysis of Agricultural Images via Dynamic Graphics. Technical Report 9/94, Laboratório Nacional de Computação Científica, Rio de Janeiro, Brazil.
- Kleinow, T. and Lehmann, H. (2002). Client/Server Based Statistical Computing. *Computational Statistics*, 17(3):315–328.
- Klinke, S. and Cook, D. (1997). Binning of Kernel-based Projection Pursuit Indices in XGobi. *Computational Statistics & Data Analysis*, 27(3):363–369.
- Klößgen, W. and Zytchow, J.M., editors (2002). *Handbook of Data Mining and Knowledge Discovery*. Oxford University Press, New York, NY.
- Kruskal, J.B. (1969). Toward a Practical Method Which Helps Uncover the Structure of a Set of Observations by Finding the Line Transformation Which Optimizes a New “Index of Condensation”. In Milton, R.C. and Nelder, J.A., editors, *Statistical Computation*, pages 427–440. Academic Press, New York, NY.
- Kruskal, J.B. (1970). Multidimensional Scaling. ASA Statistical Graphics Video Lending Library (contact: dj@research.bell-labs.com).
- Maar, D. (1982). *Vision*. Freeman, New York, NY.
- MacDougall, E.B. (1992). Exploratory Analysis, Dynamic Statistical Visualization, and Geographic Information Systems. *Cartography and Geographic Information Systems*, 19(4):237–246.
- Macedo, M., Cook, D., and Brown, T.J. (2000). Visual Data Mining in Atmospheric Science Data. *Data Mining and Knowledge Discovery*, 4(1):69–80.
- MathSoft (1996). *S+GISLink*. MathSoft, Inc., Seattle, WA.
- McDonald, J.A. and Willis, S. (1987). Use of the Grand Tour in Remote Sensing. ASA Statistical Graphics Video Lending Library (contact: dj@research.bell-labs.com).
- Monmonier, M. (1988). Geographical Representations in Statistical Graphics: A Conceptual Framework. In *1988 Proceedings of the Section on Statistical Graphics*, pages 1–10, Alexandria, VA. American Statistical Association.
- Monmonier, M. (1989). Geographic Brushing: Enhancing Exploratory Analysis of the Scatterplot Matrix. *Geographical Analysis*, 21(1):81–84.

- Morton, S.C. (1989). Interpretable Projection Pursuit. Technical Report 106, Laboratory for Computational Statistics, Stanford University.
- Morton, S.C. (1992). Interpretable Exploratory Projection Pursuit. In Page, C. and LePage, R., editors, *Proceedings of the 22nd Symposium on the Interface between Computing Science and Statistics*, pages 470–474. New York, NY. Springer.
- Murdoch, D.J. (2002). Drawing a Scatterplot. *Chance*, 13(3):53–55.
- Nagel, M. (1994). Interactive Analysis of Spatial Data. In Dirschedl, P. and Ostermann, R., editors, *Computational Statistics*, pages 295–314. Physica-Verlag, Heidelberg.
- Nagel, M., Benner, A., Ostermann, R., and Henschke, K. (1996). *Grafische Datenanalyse*. Gustav Fischer, Stuttgart; in German.
- Nelson, L., Cook, D., and Cruz-Neira, C. (1998). XGobi vs the C2: An Experiment Comparing Data Visualization in an Immersive Virtual Environment with a Workstation Display. In Cruz-Neira, C. and Riedel, O., editors, *2nd International Immersive Projection Technology Workshop, May 11–12, 1998, Iowa State University, Ames, IA*, (CD).
- Nelson, L., Cook, D., and Cruz-Neira, C. (1999). XGobi vs the C2: Results of an Experiment Comparing Data Visualization in a 3-D Immersive Virtual Reality Environment with a 2-D Workstation Display. *Computational Statistics: Special Issue on Interactive Graphical Data Analysis*, 14(1):39–51.
- Openshaw, S. and Perrée, T. (1996). User-Centred Intelligent Spatial Analysis of Point Data. In Parker, D., editor, *Innovations in GIS 3*, pages 119–134. Taylor & Francis, London, U.K.
- Ostermann, R. and Nagel, M. (1993). Dynamic Graphics for Discrete Data. *Computational Statistics*, 8:197–205.
- Pimentel, K. and Teixeira, K. (1995). *Virtual Reality through the New Looking Glass (Second Edition)*. McGraw-Hill, New York, NY.
- Posse, C. (1995). Tools for Two-dimensional Exploratory Projection Pursuit. *Journal of Computational and Graphical Statistics*, 4(2):83–100.
- Rao, C.R., editor (1993). *Handbook of Statistics, Vol. 9: Computational Statistics*. North Holland/Elsevier Science Publishers, Amsterdam.
- Rollmann, W. (1853a). Notiz zur Stereoskopie. *Annalen der Physik und Chemie*, 89:350–351; in German.
- Rollmann, W. (1853b). Zwei neue stereoskopische Methoden. *Annalen der Physik und Chemie*, 90:186–187; in German.
- Rösch, S. (1954). 100 Jahre Anaglyphen. *Die Farbe*, 3(1/2):1–6; in German.
- Rossini, A.J. and West, R.W. (1998). Virtual Reality and Statistical Research. *Computing Science and Statistics*, 29(1):215–219.
- Roy, T., Cruz-Neira, C., and DeFanti, T.A. (1995). Cosmic Worm in the CAVE: Steering a High Performance Computing Application from a Virtual Environment. *Presence: Teleoperators and Virtual Environments*, 4(2):121–129.
- Schneider, M., Stamm, C., Symanzik, J., and Widmayer, P. (2000). Virtual Reality and Dynamic Statistical Graphics: A Bidirectional Link in a Heterogeneous, Distributed Computing Environment. In *Proceedings of the International*

- Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA'2000), Las Vegas, Nevada, June 26–29, 2000, Volume IV*, pages 2345–2351. CSREA Press.
- Scott, D.W. (1992). *Multivariate Density Estimation: Theory, Practice, and Visualization*. Wiley, New York, NY.
- Scott, L.M. (1994). Identification of a GIS Attribute Error Using Exploratory Data Analysis. *The Professional Geographer*, 46(3):378–386.
- Soukop, T. and Davidson, I. (2002). *Visual Data Mining*. Wiley, New York, NY.
- Stuetzle, W. (1988). Plot Windows. In Cleveland, W.S. and McGill, M.E., editors, *Dynamic Graphics for Statistics*, pages 225–245. Wadsworth & Brooks/Cole, Belmont, CA.
- Swayne, D.F. and Buja, A. (1998). Missing Data in Interactive High-Dimensional Data Visualization. *Computational Statistics: Special Issue on Strategies for Data Analysis*, 13(1):15–26.
- Swayne, D.F. and Cook, D. (1992). Xgobi: A Dynamic Graphics Program Implemented in X With a Link to S. *Computing Science and Statistics*, 22:544–547.
- Swayne, D.F., Cook, D., and Buja, A. (1991). XGobi: Interactive Dynamic Graphics in the X Window System with a Link to S. In *1991 Proceedings of the Section on Statistical Graphics*, pages 1–8, Alexandria, VA. American Statistical Association.
- Swayne, D.F., Cook, D., and Buja, A. (1998). XGobi: Interactive Dynamic Graphics in the X Window System. *Journal of Computational and Graphical Statistics*, 7(1):113–130.
- Swayne, D.F., Temple Lang, D., Buja, A., and Cook, D. (2003). GGobi: Evolving from XGobi into an Extensible Framework for Interactive Data Visualization. *Computational Statistics & Data Analysis: Special Issue on Data Visualization*, 43(4):423–444.
- Symanzik, J. (1992). Computerdarstellungen von Anaglyphen – Ein Hilfsmittel der multivariaten Statistik. Diplomarbeit, Fachbereich Informatik, Universität Dortmund; in German.
- Symanzik, J. (1993a). Anaglyphen 3D – A Program for the Interactive Representation of Three-Dimensional Perspective Plots of Statistical Data. In Opitz, O., Lausen, B., and Klar, R., editors, *Information and Classification. Concepts, Methods and Applications. Proceedings of the 16th Annual Conference of the "Gesellschaft für Klassifikation e. V."*, pages 384–389, Berlin, Heidelberg. Springer.
- Symanzik, J. (1993b). Three-Dimensional Statistical Graphics based on Interactively Animated Anaglyphs. In *1993 Proceedings of the Section on Statistical Graphics*, pages 71–76, Alexandria, VA. American Statistical Association.
- Symanzik, J., Ascoli, G.A., Washington, S.S., and Krichmar, J.L. (1999a). Visual Data Mining of Brain Cells. *Computing Science and Statistics*, 31:445–449.
- Symanzik, J., Axelrad, D.A., Carr, D.B., Wang, J., Wong, D., and Woodruff, T.J. (1999b). HAPs, Micromaps and GPL – Visualization of Geographically Referenced Statistical Summaries on the World Wide Web. In *Annual Proceedings (ACSM-WFPS-PLSO-LSAW 1999 Conference CD)*. American Congress on Surveying and Mapping.

- Symanzik, J., Carr, D.B., Axelrad, D.A., Wang, J., Wong, D., and Woodruff, T.J. (1999c). Interactive Tables and Maps – A Glance at EPA’s Cumulative Exposure Project Web Page. In *1999 Proceedings of the Section on Statistical Graphics*, pages 94–99, Alexandria, VA. American Statistical Association.
- Symanzik, J., Cook, D., Klinke, S., and Lewin, N. (1998a). Exploration of Satellite Images in the Dynamically Linked ArcView/XGobi/XploRe Environment. In Bodt, B.A., editor, *Proceedings of the Third Annual U.S. Army Conference on Applied Statistics, 22–24 October 1997*, pages 23–33, Aberdeen Proving Ground, MD. Army Research Laboratory ARL-SR-74.
- Symanzik, J., Cook, D., Kohlmeyer, B.D., and Cruz-Neira, C. (1996a). Dynamic Statistical Graphics in the CAVE Virtual Reality Environment. Working Paper for the Dynamic Statistical Graphics Workshop, Sydney, July 7, 1996. Department of Statistics, Iowa State University, Ames, IA, Preprint 96–10, available at http://www.math.usu.edu/~symanzik/papers/1996_dsg.pdf.
- Symanzik, J., Cook, D., Kohlmeyer, B.D., Lechner, U., and Cruz-Neira, C. (1997). Dynamic Statistical Graphics in the C2 Virtual Reality Environment. *Computing Science and Statistics*, 29(2):41–47.
- Symanzik, J., Cook, D., Lewin-Koh, N., Majure, J.J., and Megretskaia, I. (2000a). Linking ArcView and XGobi: Insight Behind the Front End. *Journal of Computational and Graphical Statistics*, 9(3):470–490.
- Symanzik, J., Hurst, J., and Gunter, L. (2002a). Recent Developments for Interactive Statistical Graphics on the Web Using “nViZn”. In *2002 Proceedings*, Alexandria, VA. American Statistical Association. (CD).
- Symanzik, J. and Jones, L. (2001). “nViZn” Federal Statistical Data on the Web. In *2001 Proceedings*, Alexandria, VA. American Statistical Association. (CD).
- Symanzik, J., Majure, J.J., and Cook, D. (1996b). Dynamic Graphics in a GIS: A Bidirectional Link between ArcView 2.0 and XGobi. *Computing Science and Statistics*, 27:299–303.
- Symanzik, J., Pajarola, R., and Widmayer, P. (1998b). XGobi and XploRe Meet ViRGIS. In *1998 Proceedings of the Section on Statistical Graphics*, pages 50–55, Alexandria, VA. American Statistical Association.
- Symanzik, J., Wegman, E.J., Braverman, A.J., and Luo, Q. (2002b). New Applications of the Image Grand Tour. *Computing Science and Statistics*, 34: (CD).
- Symanzik, J., Wong, D., Wang, J., Carr, D.B., Woodruff, T.J., and Axelrad, D.A. (2000b). Web-based Access and Visualization of Hazardous Air Pollutants. In *Geographic Information Systems in Public Health: Proceedings of the Third National Conference August 18–20, 1998, San Diego, California*. Agency for Toxic Substances and Disease Registry. <http://www.atsdr.cdc.gov/GIS/conference98/>.
- Theus, M. (1996). *Theorie und Anwendung Interaktiver Statistischer Graphik*. Dr. Bernd Wißner, Augsburg; in German.
- Theus, M. (2002). Interactive Data Visualization Using Mondrian. *Journal of Statistical Software*, 7(11). <http://www.jstatsoft.org/v07/i11/>.
- Theus, M. (2003). Abstract: Interactive Data Visualization Using Mondrian. *Journal of Computational and Graphical Statistics*, 12(1):243–244.

- Theus, M., Hofmann, H., and Wilhelm, A.F.X. (1998). Selection Sequences – Interactive Analysis of Massive Data Sets. *Computing Science and Statistics*, 29(1):439–444.
- Theus, M. and Wilhelm, A.F.X. (1998). Counts, Proportions, Interactions – A View on Categorical Data. In *1998 Proceedings of the Section on Statistical Graphics*, pages 6–15, Alexandria, VA. American Statistical Association.
- Tukey, J.W. (1977). *Exploratory Data Analysis*. Addison Wesley, Reading, MA.
- Tweedie, L. and Spence, R. (1998). The Projection Matrix: A Tool to Support the Interactive Exploration of Statistical Models and Data. *Computational Statistics: Special Issue on Strategies for Data Analysis*, 13(1):65–76.
- Unwin, A. (1994). REGARDing Geographic Data. In Dirschedl, P. and Ostermann, R., editors, *Computational Statistics*, pages 315–326. Physica-Verlag, Heidelberg.
- Unwin, A. (1999). Requirements for Interactive Graphics Software for Exploratory Data Analysis. *Computational Statistics: Special Issue on Interactive Graphical Data Analysis*, 14(1):7–22.
- Unwin, A. (2002). Scatterplotting. *Chance*, 15(2):39–42.
- Unwin, A., Hawkins, G., Hofmann, H., and Siegl, B. (1996). Interactive Graphics for Data Sets with Missing Values – MANET. *Journal of Computational and Graphical Statistics*, 5(2):113–122.
- Unwin, A. and Wills, G. (1988). Eyeballing Time Series. In *1988 Proceedings of the Section on Statistical Computing*, pages 263–268, Alexandria, VA. American Statistical Association.
- Unwin, A., Wills, G., and Haslett, J. (1990). REGARD – Graphical Analysis of Regional Data. In *1990 Proceedings of the Section on Statistical Graphics*, pages 36–41, Alexandria, VA. American Statistical Association.
- Valero-Mora, P.M., Young, F.W., and Friendly, M. (2003). Visualizing Categorical Data in ViSta. *Computational Statistics & Data Analysis: Special Issue on Data Visualization*, 43(4):495–508.
- van Teylingen, R., Ribarsky, W., and van der Mast, C. (1997). Virtual Data Visualizer. *IEEE Transactions on Visualization and Computer Graphics*, 3(1):65–74.
- Vince, J. (1995). *Virtual Reality Systems*. ACM Press/Addison-Wesley, Wokingham, UK.
- Vuibert, H. (1912). *Les Anaglyphes Géométriques*. Librairie Vuibert, Paris; in French.
- Wang, X., Chen, J.X., Carr, D.B., Bell, B.S., and Pickle, L.W. (2002). Geographic Statistics Visualization: Web-based Linked Micromap Plots. *Computing in Science & Engineering*, 4(3):90–94.
- Wegman, E.J. (1990). Hyperdimensional Data Analysis Using Parallel Coordinates. *Journal of the American Statistical Association*, 85:664–675.
- Wegman, E.J. (1992). The Grand Tour in k-Dimensions. *Computing Science and Statistics*, 22:127–136.
- Wegman, E.J. (1995). Huge Data Sets and the Frontiers of Computational Feasibility. *Journal of Computational and Graphical Statistics*, 4(4):281–295.

- Wegman, E.J. (2000). Visions: New Techniques and Technologies in Statistics. *Computational Statistics: Special Issue on New Techniques and Technologies for Statistics*, 15(1):133–144.
- Wegman, E.J. and Carr, D.B. (1993). Statistical Graphics and Visualization. In Rao, C.R., editor, *Handbook of Statistics, Vol. 9: Computational Statistics*, pages 857–958. North Holland/Elsevier Science Publishers, Amsterdam.
- Wegman, E.J. and DePriest, D.J., editors (1986). *Statistical Image Processing and Graphics*. Marcel Dekker, New York, NY.
- Wegman, E.J. and Dorfman, A. (2003). Visualizing Cereal World. *Computational Statistics & Data Analysis: Special Issue on Data Visualization*, 43(4):633–649.
- Wegman, E.J. and Luo, Q. (1997a). High Dimensional Clustering Using Parallel Coordinates and the Grand Tour. *Computing Science and Statistics*, 28:361–368.
- Wegman, E.J. and Luo, Q. (1997b). High Dimensional Clustering Using Parallel Coordinates and the Grand Tour. In Klar, R. and Opitz, O., editors, *Classification and Knowledge Organization*, pages 93–101. Springer.
- Wegman, E.J., Poston, W.L., and Solka, J.L. (1998). Image Grand Tour. In *Automatic Target Recognition VIII – Proceedings of SPIE, 3371, 286–294*; republished in: Sadjadi, F. (Ed.), Vol. 6: *Automatic Target Recognition. The CD-ROM*, Bellingham, WA, 1999. SPIE.
- Wegman, E.J. and Shen, J. (1993). Three-Dimensional Andrews Plots and the Grand Tour. *Computing Science and Statistics*, 25:284–288.
- Wegman, E.J. and Symanzik, J. (2002). Immersive Projection Technology for Visual Data Mining. *Journal of Computational and Graphical Statistics*, 11(1):163–188.
- Wegman, E.J., Symanzik, J., Vandersluis, J.P., Luo, Q., Camelli, F., Dzubay, A., Fu, X., Khumbah, N.-A., Moustafa, R. E.A., Wall, R.L., and Zhu, Y. (1999). The MiniCAVE – A Voice-Controlled IPT Environment. In Bullinger, H.-J. and Riedel, O., editors, *3. International Immersive Projection Technology Workshop, 10./11. May 1999, Center of the Fraunhofer Society Stuttgart IZS*, pages 179–190, Berlin, Heidelberg. Springer.
- Wilhelm, A.F.X., Unwin, A., and Theus, M. (1996). Software for Interactive Statistical Graphics – A Review. In Faulbaum, F. and Bandilla, W., editors, *SoftStat '95 – Advances in Statistical Software 5*, pages 3–12, Stuttgart. Lucius & Lucius.
- Wilhelm, A.F.X., Wegman, E.J., and Symanzik, J. (1999). Visual Clustering and Classification: The Oronsay Particle Size Data Set Revisited. *Computational Statistics: Special Issue on Interactive Graphical Data Analysis*, 14(1):109–146.
- Wilkinson, L. (1999). *The Grammar of Graphics*. Springer, New York, NY.
- Wilkinson, L., Rope, D.J., Carr, D.B., and Rubin, M.A. (2000). The Language of Graphics. *Journal of Computational and Graphical Statistics*, 9(3):530–543.
- Williams, I., Limp, W.F., and Briuer, F.L. (1990). Using Geographic Information Systems and Exploratory Data Analysis for Archaeological Site Classification and Analysis. In Allen, K. M.S., Green, S.W., and Zubrow, E. B.W., editors, *Interpreting Space: GIS and Archaeology*, pages 239–273. Taylor & Francis, London, U.K.
- Witten, I.H. and Frank, E. (2000). *Data Mining – Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufmann/Academic Press, San Francisco, CA/San Diego, CA.

- Wojciechowski, W.C. and Scott, D.W. (2000). High-Dimensional Visualization using Continuous Conditioning. *Computing Science and Statistics*, 32. (CD).
- Yoshioka, K. (2002). KyPlot – A User-Oriented Tool for Statistical Data Analysis and Visualization. *Computational Statistics*, 17(3):425–437.
- Young, F.W., Faldowski, R.A., and McFarlane, M.M. (1993). Multivariate Statistical Visualization. In Rao, C.R., editor, *Handbook of Statistics, Vol. 9: Computational Statistics*, pages 959–998. North Holland/Elsevier Science Publishers, Amsterdam.
- Zhang, Z. and Griffith, D.A. (1997). Developing User-Friendly Spatial Statistical Analysis Modules for GIS: An Example using ArcView. *Computers, Environment and Urban Systems*, 21(1):5–29.

The Grammar of Graphics

II.11

Leland Wilkinson

11.1	<i>Introduction</i>	339
	Architecture	339
11.2	<i>Variables</i>	340
	Variable	341
	Varset	342
	Converting a Table of Data to a Varset	342
11.3	<i>Algebra</i>	343
	Operators	343
	Rules	351
	SQL Equivalences	351
	Related Algebras	352
	Algebra XML	353
11.4	<i>Scales</i>	353
	Axiomatic Measurement	354
	Unit Measurement	354
	Transformations	355
11.5	<i>Statistics</i>	357
11.6	<i>Geometry</i>	358
11.7	<i>Coordinates</i>	362
11.8	<i>Aesthetics</i>	363
11.9	<i>Layout</i>	366
11.10	<i>Analytics</i>	367
	Statistical Model Equivalents	367
	Subset Model Fitting	370
	Lack of Fit	370

Scalability	371
An Example.....	371
<i>11.11 Software</i>	373
<i>11.12 Conclusion</i>	375

Introduction

11.1

The Grammar of Graphics, or GOG, denotes a system with seven orthogonal components (Wilkinson, 1999). By *orthogonal*, we mean there are seven graphical component sets whose elements are aspects of the general system and that every combination of aspects in the product of all these sets is meaningful. This sense of the word orthogonality, a term used by computer designers to describe a combinatoric system of components or building blocks, is in some sense similar to the orthogonal factorial analysis of variance (ANOVA), where factors have levels and all possible combinations of levels exist in the ANOVA design. If we interpret each combination of features in a GOG system as a point in a network, then the world described by GOG is represented in a seven-dimensional rectangular lattice.

A consequence of the orthogonality of such a graphic system is a high degree of *expressiveness*. That is, it comprises a system that can produce a huge variety of graphical forms (chart types). In fact, it is claimed that virtually the entire corpus of known charts can be generated by this relatively parsimonious system, and perhaps a great number of meaningful but undiscovered chart types as well.

The second principal claim of GOG is that this system describes the meaning of what we do when we construct statistical graphs or charts. It is more than a taxonomy. It is a computational system based on the classical mathematics of representing functions and relations in Cartesian and other spaces. Because of this mathematical foundation, GOG specifications can serve as parsimonious and natural descriptions of famous statistical charts devised by Playfair, Minard, Jevons, Pearson, Bertin, Tukey, and other significant figures in the history of statistical graphics.

Architecture

11.1.1

Figure 11.1 shows a *dataflow* diagram that contains the seven GOG components. This dataflow is a chain that describes the sequence of mappings needed to produce a statistical graphic from a set of data. The first component (Variables) maps data to an object called a *varset* (a set of variables). The next three components (Algebra, Scales, Statistics) are transformations on varsets. The next component (Geometry) maps a varset to a graph and the next (Coordinates) embeds a graph in a coordinate space. The last component (Aesthetics) maps a graph to a visible or perceivable display called a graphic.

The dataflow architecture implies that the subtasks needed to produce a graphic from data must be done in this specified order. Imposing an order would appear to be unnecessarily restrictive, but changes of this ordering can produce meaningless graphics. For example, if we compute certain statistics on variables (e.g., sums) before scaling them (e.g., log scales), we can produce statistically questionable results because the log of a sum is not the sum of the logs.

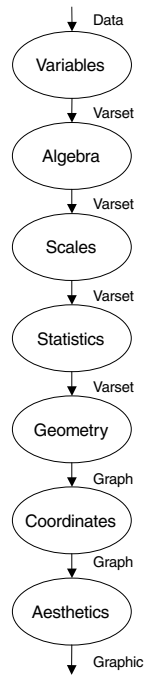


Figure 11.1. Dataflow

The dataflow in Fig. 11.1 has many paths through it. We can choose different designs (factorial, nested, ...), scales (log, probability, ...), statistical methods (means, medians, modes, smoothers, ...), geometric objects (points, lines, bars, ...), coordinate systems (rectangular, polar, ...), and aesthetics (size, shape, color, ...). These paths reveal the richness of the system. The remainder of this article will summarize the seven GOG components, delineate these paths, and then briefly introduce sample applications.

11.2 Variables

We begin with data. We assume the data that we wish to graph are organized in one or more tables. The column(s) of each table represent a set of fields, each field containing a set of measurements or attributes. The row(s) of this table represent a set of logical records, each record containing the measurements of an object on each field. Usually, a relational database management system (RDBMS) produces such a table from organized queries specified in Structured Query Language (SQL) or another relational language. When we do not have data stored in a relational database (e.g., live data feeds), we need custom software to provide such a table using Extensible Markup Language (XML), Perl scripts, or other languages.

The first thing we have to do is convert such tables of data to something called a *varset*. A *varset* is a set of one or more variables. While a column of a table of data might superficially be considered to be a variable, there are differences. A variable is both more general (in regard to generalizability across samples) and more specific (in regard to data typing and other constraints) than a column of data. First, we define a variable, then a *varset*.

Variable

11.2.1

A statistical *variable* X is a mapping $f : O \rightarrow V$, which we consider as a triple:

$$X = [O, V, f]$$

The domain O is a set of objects.

The codomain V is a set of values.

The function f assigns to each element of O an element in V .

The image of O under f contains the *values* of X . We denote a possible value as x , where $x \in V$. We denote a value of an object as $X(o)$, where $o \in O$. A variable is *continuous* if V is an interval. A variable is *categorical* if V is a finite subset of the integers (or there exists an injective map from V to a finite subset of the integers).

Variables may be multidimensional. X is a p -dimensional variable made up of p one-dimensional variables:

$$\begin{aligned} \mathbf{X} &= (X_1, \dots, X_p) \\ &= [O, V_i, f], \quad i = 1, \dots, p \\ &= [O, \mathbf{V}, f] \end{aligned}$$

The element $\mathbf{x} = (x_1, \dots, x_p)$, $\mathbf{x} \in \mathbf{V}$, is a p -dimensional value of \mathbf{X} . We use multidimensional variables in *multivariate analysis*.

A *random variable* X is a real-valued function defined on a sample space Ω :

$$X = [\Omega, \mathbb{R}, f]$$

Random variables may be multidimensional. In the elementary probability model, each element $\omega \in \Omega$ is associated with a probability function P . The range of P is the interval $[0, 1]$, and $P(\Omega) = 1$. Because of the associated probability model, we can make probability statements about outcomes in the range of the random variable, such as:

$$P(X = 2) = P(\omega : \omega \in \Omega, X(\omega) = 2)$$

11.2.2 Varset

We call the triple

$$X = [V, \tilde{O}, f]$$

a varset. The word *varset* stands for *variable set*. If X is multidimensional, we use boldface X . A varset inverts the mapping used for variables. That is,

The domain V is a set of values.

The codomain \tilde{O} is a set of all possible ordered lists of objects.

The function f assigns to each element of V an element in \tilde{O} .

We invert the mapping customarily used for variables in order to simplify the definitions of graphics algebra operations on varsets. In doing so, we also replace the variable's set of objects with the varset's set of ordered lists. We use lists in the codomain because it is possible for a value to be mapped to an object more than once (as in repeated measurements).

11.2.3 Converting a Table of Data to a Varset

To convert a table to a varset, we must define the varset's domain of values and range of objects and specify a reference function that maps each row in the table to an element in the varset.

We define the domain of values in each varset by identifying the measurements its values represent. If our measurements include weight, for example, we need to record the measurement units and the interval covered by the range of weights. If the domain is categorical, we need to decide whether there are categories not found in the data that should be included in the domain (refused to reply, don't know, missing, ...). And we may need to identify categories found in the data that are not defined in the domain (mistakes, intransitivities, ...). The varset's domain is a key component in the GOG system. It is used for axes, legends, and other components of a graphic. Because the actual data for a chart are only an instance of what we expect to see in a varset's domain, we let the domain control the structure of the chart.

We define the range of potential objects in each varset by identifying the class they represent. If the rows of our table represent measurements of a group of school children, for example, we may define the range to be school children, children in general, people, or (at the most abstract level) objects. Our decision about the level of generality may affect how a graphic will be titled, how legends are designed, and so on.

Finally, we devise a reference system by indexing objects in the domain. This is usually as simple as deriving a *caseID* from a table row index. Or, as is frequently done, we may choose the value of a key variable (e.g., Social Security Number) to create a unique index.

Algebra

11.3

Given one or more varsets, we now need to operate on them to produce combinations of variables. A typical scatterplot of a variable X against a variable Y , for example, is built from tuples (x_i, y_i) that are elements in a set product. We use graphics algebra on values stored in varsets to make these tuples. There are three binary operators in this algebra: *cross*, *nest*, and *blend*.

Operators

11.3.1

We will define these operators in set notation and illustrate them by using a table of real data. Table 11.1 shows 1980 and 2000 populations for selected world cities. During various periods in US history, it was fashionable to name towns and

Table 11.1. Cities and their Populations

Country	City	1980 Population	2000 Population
Japan	Tokyo	21,900,000	26,400,000
India	Mumbai	8,067,000	18,100,000
USA	New York	15,600,000	16,600,000
Nigeria	Lagos	4,385,000	13,400,000
USA	Los Angeles	9,523,000	13,100,000
Japan	Osaka	9,990,000	11,000,000
Philippines	Manila	5,955,000	10,900,000
France	Paris	8,938,000	9,624,000
Russia	Moscow	8,136,000	9,321,000
UK	London	7,741,000	7,640,000
Peru	Lima	4,401,000	7,443,000
USA	Chicago	6,780,000	6,951,000
Iraq	Bagdad	3,354,000	4,797,000
Canada	Toronto	3,008,000	4,651,000
Spain	Madrid	4,296,000	4,072,000
Germany	Berlin	3,247,000	3,324,000
Australia	Melbourne	2,765,000	3,187,000
USA	Melbourne	46,536	71,382
USA	Moscow	16,513	21,291
USA	Berlin	13,084	10,331
USA	Paris	9885	9077
USA	London	4002	5692
USA	Toronto	6934	5676
USA	Manila	2553	3055
USA	Lima	2025	2459
USA	Madrid	2281	2264
USA	Bagdad	2331	1578

cities after their European and Asian counterparts. Sometimes this naming was driven by immigration, particularly in the colonial era (New Amsterdam, New York, New London). At other times, exotic names reflected a fascination with foreign travel and culture, particularly in the Midwest (Paris, Madrid). Using a dataset containing namesakes will help reveal some of the subtleties of graphics algebra.

We begin by assuming there are four varsets derived from this table: `country`, `city`, `pop1980`, and `pop2000` (we use lower case for varsets when they are denoted by names instead of single letters). Each varset has one column. The varsets resulting from algebraic operations will have one or more columns.

There is one set of objects for all four varsets: 27 cities. This may or may not be a subset of the domain for the four associated variables. If we wish to generalize analyses of this varset to other cities, then the set of possible objects in these varsets might be a subdomain of the set of all cities existing in 1980 and 2000. We might even consider this set of objects to be a subset of all possible cities in all of recorded history. While these issues might seem more the province of sampling and generalizability theory, they affect the design of a graphics system. Databases, for example, include facilities for *semantic integrity constraints* that ensure domain integrity in data tables. Data-based graphics systems share similar requirements.

There are sets of values for these varsets. The `country` varset has country names in the set of values comprising its domain. The definition of the domain of the varset depends on how we wish to use it. For example, we might include spellings of city names in languages other than English. We might also include country names not contained in this particular varset. Such definitions would affect whether we could add new cities to a database containing these data. For `pop1980` and `pop2000`, we would probably make the domain be the set of positive integers.

Cross (*)

Cross joins the left argument with the right to produce a set of tuples stored in the multiple columns of the new varset:

$$\begin{array}{|c|} \hline x \\ \hline y \\ \hline z \\ \hline \end{array} * \begin{array}{|c|} \hline a \\ \hline a \\ \hline b \\ \hline \end{array} = \begin{array}{|c|c|} \hline x & a \\ \hline y & a \\ \hline z & b \\ \hline \end{array}$$

The resulting set of tuples is a subset of the product of the domains of the two varsets. The domain of a varset produced by a cross is the product of the separate domains.

One may think of a cross as a horizontal concatenation of the table representation of two varsets, assuming the rows of each varset are equivalent and in the same order. The following example shows a crossing of two varsets using set notation with simple integer keys for the objects:

$$A = [\{red, blue\}, \{\langle \cdot \rangle, \langle \cdot, \cdot \rangle, \dots\}, \{red \rightarrow \langle 1, 4 \rangle, blue \rightarrow \langle 2, 3 \rangle\}]$$

$$B = [[-10, 10], \{\langle \cdot \rangle, \langle \cdot, \cdot \rangle, \dots\}, \{-10 \rightarrow \langle 1 \rangle, 5 \rightarrow \langle 2, 3 \rangle, 10 \rightarrow \langle 4 \rangle\}]$$

$$A * B = [\{red, blue\} \times [-10, 10], \{\langle \cdot \rangle, \langle \cdot, \cdot \rangle, \dots\},$$

$$\{(red, -10) \rightarrow \langle 1 \rangle, (blue, 5) \rightarrow \langle 2, 3 \rangle, (red, 10) \rightarrow \langle 4 \rangle\}]$$

If we plotted $A * B$ in two dimensions with a point graph, we would see n points between -10 and 10 stacked vertically above one or both of the two color names.

Figure 11.2 shows a graphic based on the algebraic expression `city * pop2000`. We choose the convention of representing the first variable in an expression on the horizontal axis and the second on the vertical. We also restrict the domain of `pop2000` to be $[0, 32,000,000]$.

Although most of the US namesake cities have smaller populations, it is not easy to discern them in the graphic. We can separate the US from the other cities by using a variable called `group` that we derive from the country names. Such a new variable is created easily in a database or statistical transformation language with an expression like

```
if (country == ``USA``) group = ``USA``;
else group = ``World``;
```

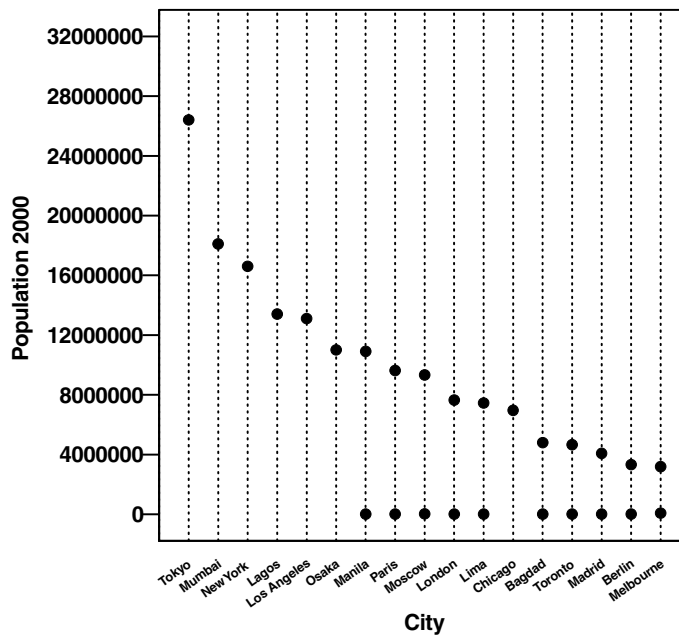


Figure 11.2. `city * pop2000`

Figure 11.3 shows a graphic based on the three-dimensional algebraic expression `city * pop2000 * group`. This expression produces a varset with three columns. The first column is assigned to the horizontal axis, the second to the vertical, and the third to the horizontal axis again, which has the effect of splitting the frame into two frames. This general pattern of alternating horizontal and vertical roles for the columns of a varset provides a simple layout scheme for complex algebraic expressions. We may think of this as a generalization of the Trellis layout scheme (Becker et al., 1996). We could, of course, represent this same varset in a 3D plot projected into 2D, but the default system behavior is to prefer 2D with recursive partitioning. We will describe this in more detail in Sect 11.9.

Chicago stands out as an anomaly in Fig. 11.3 because of its relatively large population. We might want to sort the cities in a different order for the left panel or eliminate cities not found in the US, but the algebraic expression won't let us do that. Because `group` is crossed with the other variables, there is only one domain of cities shared by both country groups. If we want to have different domains for the two panels, we need our next operator, *nest*.

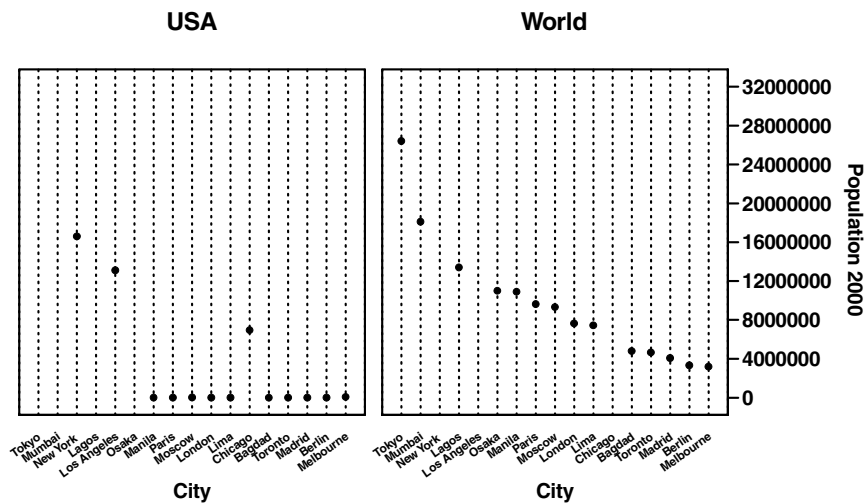


Figure 11.3. `city * pop2000 * group`

Nest (/)

Nest partitions the left argument using the values in the right:

$$\begin{array}{|c|} \hline x \\ \hline y \\ \hline z \\ \hline \end{array} / \begin{array}{|c|} \hline a \\ \hline a \\ \hline b \\ \hline \end{array} = \begin{array}{|c|c|} \hline x & a \\ \hline y & a \\ \hline z & b \\ \hline \end{array}$$

Although it is not required in the definition, we assume the nesting varset on the right is categorical. If it were continuous (having interval domain) there would be an infinite number of partitions. We do require predefined nested domains. To construct a nested domain, three options are possible:

1. Data values – identify the minimal domain containing the data by enumerating unique data tuples.
2. Metadata – define the domain using external rules contained in a metadata resource or from known principles.
3. Data organization – identify nested domains using the predefined structure of a hierarchical database or OLAP cube.

The following example shows a nesting of two categorical variables:

$$A = [\{ant, fly, bee\}, \{\langle \cdot \rangle, \langle \cdot, \cdot \rangle, \dots\}, \{ant \rightarrow \langle 1 \rangle, fly \rightarrow \langle 2, 3 \rangle, bee \rightarrow \langle 4 \rangle\}]$$

$$B = [\{noun, verb\}, \{\langle \cdot \rangle, \langle \cdot, \cdot \rangle, \dots\}, \{noun \rightarrow \langle 1, 2, 4 \rangle, verb \rightarrow \langle 3 \rangle\}]$$

$$A/B = [\{(ant, noun), (fly, noun), (fly, verb), (bee, noun)\}, \{\langle \cdot \rangle, \langle \cdot, \cdot \rangle, \dots\}, \\ \{(ant, noun) \rightarrow \langle 1 \rangle, (fly, noun) \rightarrow \langle 2 \rangle, (fly, verb) \rightarrow \langle 3 \rangle, (bee, noun) \rightarrow \langle 4 \rangle\}]$$

Nesting defines meaning conditionally. In this example, the meaning of *fly* is ambiguous unless we know whether it is a noun or a verb. Furthermore, there is no verb for *ant* or *bee* in the English language, so the domain of A/B does not include this combination.

If A is a continuous variable, then we have something like the following:

$$A = [[0, 10], \{\langle \cdot \rangle, \langle \cdot, \cdot \rangle, \dots\}, \{0 \rightarrow \langle 1 \rangle, 8 \rightarrow \langle 2 \rangle, 1.4 \rightarrow \langle 3 \rangle, 3 \rightarrow \langle 4 \rangle, 10 \rightarrow \langle 5, 6 \rangle\}]$$

$$B = [\{1, 2\}, \{\langle \cdot \rangle, \langle \cdot, \cdot \rangle, \dots\}, \{1 \rightarrow \langle 1, 2, 3 \rangle, 2 \rightarrow \langle 4, 5, 6 \rangle\}]$$

$$A/B = [\{[0, 8] \times \{1\}, [3, 10] \times \{2\}\}, \{\langle \cdot \rangle, \langle \cdot, \cdot \rangle, \dots\}, \\ \{(0, 1) \rightarrow \langle 1 \rangle, (8, 1) \rightarrow \langle 2 \rangle, (1.4, 1) \rightarrow \langle 3 \rangle, (3, 2) \rightarrow \langle 4 \rangle, (10, 2) \rightarrow \langle 5, 6 \rangle\}]$$

In this example, the elements of the nesting A/B result in intervals conditioned on the values of B . A represents 6 ratings (ranging from 0 to 10) of the behavior of patients by two psychiatrists. B represents the identity of the psychiatrist making each rating. The intervals $[0, 8]$ and $[3, 10]$ imply that psychiatrist 1 will not use a rating greater than 8 and psychiatrist 2 will not use a rating less than 3. Nesting in this case is based on the (realistic) assumption that the two psychiatrists assign numbers to their perceptions in a different manner. A rating of 2 by one psychiatrist cannot be compared to the same rating by the other, because of possible differences in location, scale, and even local nonlinearities. Much of psychometrics is concerned with the problem of equating ratings in this type of example so that nesting would not be needed, although it is not always possible to do so plausibly.

The name *nest* comes from design-of-experiments terminology. We often use the word *within* to describe its effect. For example, if we assess schools and teachers in a district, then *teachers within schools* specifies that teachers are nested within schools. Assuming each teacher in the district teaches at only one school, we would conclude that if our data contain two teachers with the same name at different schools, they are different people. Those familiar with experimental design may recognize that the expression A/B is equivalent to the notation $A(B)$ in a design specification. Both expressions mean *A is nested within B*. Statisticians' customary use of parentheses to denote nesting conceals the fact that nesting involves an operator, however. Because nesting is distributive over blending, we have made this operator explicit and retained the conventional mathematical use of parentheses in an algebra.

Figure 11.4 shows a graphic based on the algebraic expression $city/group * pop2000$. The horizontal axis in each panel now shows a different set of cities: one for the USA and one for the rest of the world. This graphic differs from the one in Fig. 11.3 not only because the axes *look* different, but also because the meanings of the cities in each panels *are* different. For example, the city named Paris appears twice in both figures. In Fig. 11.3, on the one hand, we assume the name Paris in the left panel is comparable to the name Paris in the right. That is, it refers to a common name (Paris) occurring in two different contexts. In Fig. 11.4, on the other hand, we assume the name Paris references two different cities. They happen to have the same name, but are not equivalent. Such distinctions are critical, but often subtle.

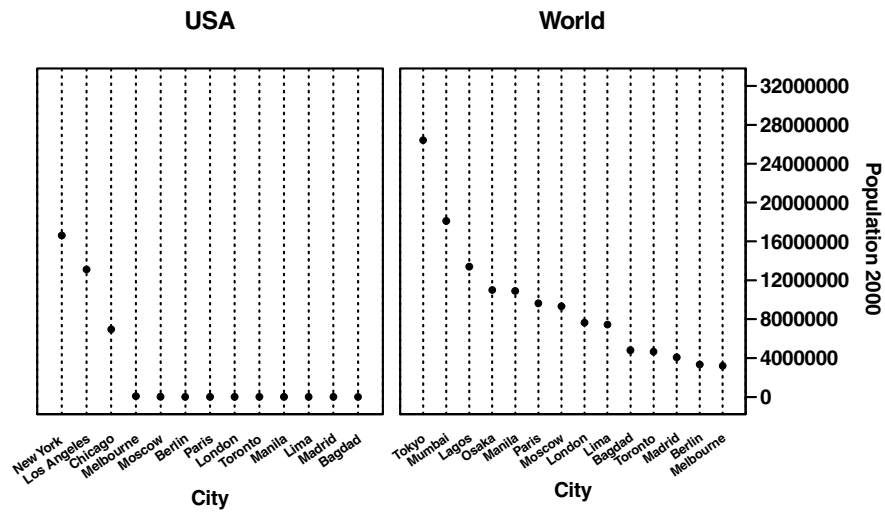


Figure 11.4. $city/group * pop2000$

Blend (+)

Blend produces a union of varsets:

$$\begin{array}{|c|} \hline x \\ \hline y \\ \hline z \\ \hline \end{array} + \begin{array}{|c|} \hline a \\ \hline a \\ \hline b \\ \hline \end{array} = \begin{array}{|c|} \hline x \\ \hline y \\ \hline z \\ \hline a \\ \hline a \\ \hline b \\ \hline \end{array}$$

Blend is defined only if the order of the tuples (number of columns) in the left and right varsets is the same. Furthermore, we should restrict blend to varsets with composable domains, even though we do not need this restriction for the operation to be defined. It would make little sense to blend Age and Weight, much less Name and Height.

In vernacular, we often use the conjunction *and* to signify that two sets are blended into one (although the word *or* would be more appropriate technically). For example, if we measure diastolic and systolic blood pressure among patients in various treatment conditions and we want to see blood pressure plotted on a common axis, we can plot diastolic *and* systolic against treatment. The following example shows a blending of two varsets, using integers for keys:

$$A = [[0, 120], \{\langle \cdot \rangle, \langle \cdot, \cdot \rangle, \dots\}, \{0 \rightarrow \langle 1 \rangle, 120 \rightarrow \langle 2 \rangle, 90 \rightarrow \langle 3, 4 \rangle\}]$$

$$B = [[10, 200], \{\langle \cdot \rangle, \langle \cdot, \cdot \rangle, \dots\}, \{10 \rightarrow \langle 1 \rangle, 200 \rightarrow \langle 2, 3 \rangle, 90 \rightarrow \langle 4 \rangle\}]$$

$$A + B = [[0, 200], \{\langle \cdot \rangle, \langle \cdot, \cdot \rangle, \dots\},$$

$$\{0 \rightarrow \langle 1 \rangle, 10 \rightarrow \langle 1 \rangle, 120 \rightarrow \langle 2 \rangle, 90 \rightarrow \langle 3, 4, 4 \rangle, 200 \rightarrow \langle 2, 3 \rangle\}]$$

Figure 11.5 shows an example of a blend using our cities data. The graphic is based on the algebraic expression `city * (pop1980 + pop2000)`. The horizontal axis represents the cities and the vertical axis represents the two repeated population measures. We have included different symbol types and a legend to distinguish the measures. We will see later how shape aesthetics are used to create this distinction.

As with the earlier graphics, we see that it is difficult to distinguish US and world cities. Figure 11.6 makes the distinction clear by splitting the horizontal axis into two nested subgroups. The graphic is based on the algebraic expression `(city/group) * (pop1980 + pop2000)`. Once again, the vertical axis represents the two repeated population measures blended on a single dimension. We see most of the cities gaining population between 1980 and 2000.

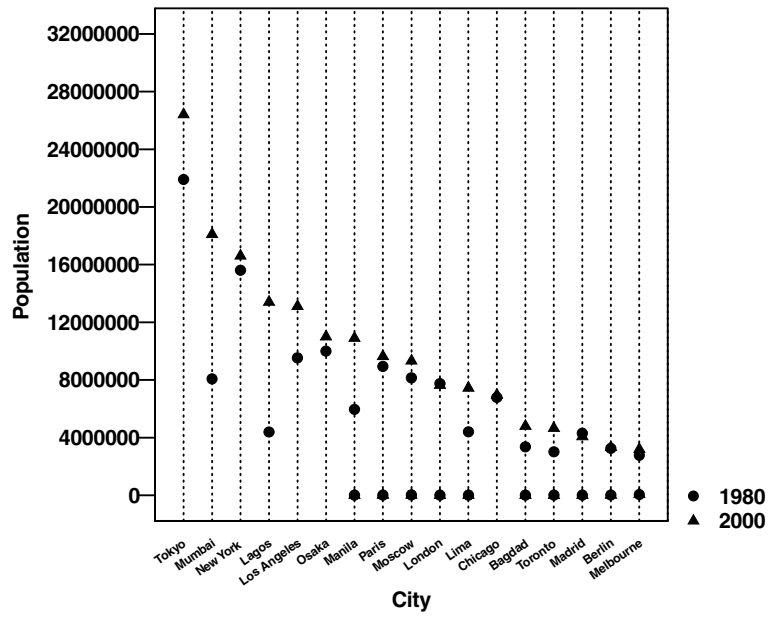


Figure 11.5. city * (pop1980 + pop2000)

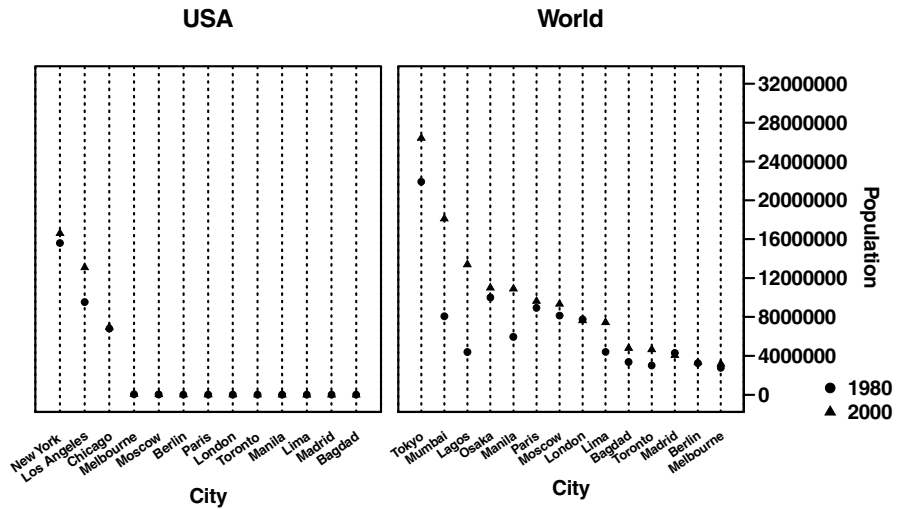


Figure 11.6. (city/group) * (pop1980 + pop2000)

Rules**11.3.2**

The following rules are derivable from the definitions of the graphics operators:

Associativity

$$(X * Y) * Z = X * (Y * Z)$$

$$(X/Y)/Z = X/(Y/Z)$$

$$(X + Y) + Z = X + (Y + Z)$$

Distributivity

$$X * (Y + Z) = X * Y + X * Z$$

$$X/(Y + Z) = X/Y + X/Z$$

$$(X + Y) * Z = X * Z + Y * Z$$

$$(X + Y)/Z = X/Z + Y/Z$$

Commutativity

$$X + Y = Y + X$$

Identity

The identity element for blend is an empty list. Cross and nest have no identity.

Precedence

Nest takes precedence over cross and blend. Cross takes precedence over blend. This hierarchical order may be altered through the use of parentheses.

SQL Equivalences**11.3.3**

Given a table X and a table Y in a database, we can use SQL to perform the operations in chart algebra. This section outlines how to do this.

Cross

Cross can be accomplished by a *cross join*:

```
SELECT a.*, b.*
FROM X a, Y b;
```

Of course, this operation is inefficient and requires optimization. Alternatively, one can do a simple *join* and generate the missing tuples with an iterator when needed.

Nest

Nest can be accomplished through a *nest* operation. The nest operator requires that the database allow tables as primitives, either as relation-valued attributes (Date and Darwen, 1992) or as nested tables (Makinouchi, 1977), (Abiteboul et al., 1989).

Alternatively, we can accumulate the subset of tuples in a nest operation with a simple *join*:

```
SELECT a.*,b.*
FROM X a,Y b
WHERE a.rowid = b.rowid;
```

If we use this latter method, we must distinguish the entries used for tags and those used for values.

Blend

Blend is performed through UNION. If UNION *all* is not available, we can concatenate key columns to be sure that all rows appear in the result set.

```
SELECT * from X
UNION all
SELECT * from Y;
```

Composition and Optimization

SQL statements can be composed by using the grammar for chart algebra. Compound statements can then be submitted for optimization and execution by a database compiler. Alternatively, pre-optimization can be performed on the chart algebra parse tree object and the optimized parse tree used to generate SQL. Secondary optimization can then be performed by the database compiler.

11.3.4 Related Algebras

Research on algebras that could be used for displaying data has occurred in many fields. We will summarize these approaches in separate sections.

Table Algebras

The US Bureau of Labor Statistics pioneered a language for laying out tables (Mendelssohn, 1974). While not a formal algebra, this Table Production Language (TPL) contained many of the elements needed to assemble complex tables. Gyssens

et al. (1996) outlined an algebra for displaying relational data; this algebra closely followed TPL, although the latter is not referenced. Wilkinson (1996) presented an algebra for structuring tables and graphics.

Design Algebras

Nelder (1965) and Wilkinson and Rogers (1973) developed a language for implementing factorial and nested experimental designs, following Fisher (1935). The operators in this language are similar to the cross and nest operators in the present paper. The algebraic design language was implemented in the GENSTAT statistical computer program for generating and analyzing general linear statistical models.

Query Algebras

Pedersen et al. (2002) described an algebra for querying OLAP cubes. The result sets from their algebraic expressions could be used for graphic displays. Agrawal et al. (1997) used a similar algebra for statistical modeling of data contained in a cube.

Display Algebras

Mackinlay (1986) developed an algebra for querying relational databases and generating charts. His general goal was to develop an intelligent system that could offer graphical responses to verbal or structural queries. Roth et al. (1994) followed a similar strategy in developing graphical representations of relational data. They extended Mackinlay's and others' ideas by using concepts from computational geometry.

Algebra XML

11.3.5

A parse tree for a given algebraic expression maps nicely to XML in a manner similar to the way MathML (<http://www.w3.org/TR/MathML2/>) is defined. We have developed an implementation, called VizML (<http://xml.spss.com/visualization>), that includes not only the algebraic components of the specification, but also the aesthetic and geometric aspects. Ultimately, VizML makes it possible to embed chart algebraic operations in a database.

Scales

11.4

Before we compute summaries (totals, means, smoothers, ...) and represent these summaries using geometric objects (points, lines, ...), we must scale our varsets. In producing most common charts, we do not notice this step. When we implement log scales, however, we notice it immediately. We must log our data before averaging logs. Even if we do not compute nonlinear transformations, however, we need to specify a measurement model.

The measurement model determines how distance in a frame region relates to the ranges of the variables defining that region. Measurement models are reflected in the axes, scales, legends, and other annotations that demarcate a chart's frame. Measurement models determine how values are represented (e.g., as categories or magnitudes) and what the units of measurement are.

11.4.1 Axiomatic Measurement

In constructing scales for statistical charts, it helps to know something about the function used to assign values to objects. Stevens (1946) developed a taxonomy of such functions based on axioms of measurement. Stevens identified four basic scale types: *nominal*, *ordinal*, *interval*, and *ratio*.

To define a nominal scale, we assume there exists at least one equivalence class together with a binary equivalence relation (\sim) that can be applied to objects in the domain (e.g., the class of this object is the *same* as the class of that object). For a domain of objects D and a set of values $X(d)$, $d \in D$, we say that a scale is nominal if

$$d_i \sim d_j \iff X(d_i) = X(d_j), \forall d_i, d_j \in D.$$

To define an ordinal scale, we assume there exists a binary total order relation ($>$) that can be applied to objects in the domain (e.g., this stone is *heavier than* that stone). We then say that a scale is ordinal if

$$d_i > d_j \iff X(d_i) > X(d_j), \forall d_i, d_j \in D.$$

To define an interval scale, we assume there exists a symmetric concatenation operation (\oplus) that can be applied to objects in the domain (e.g., the length of this stick *appended to* the length of that stick). We then say that a scale is interval if

$$d_i \oplus d_j \sim d_k \iff X(d_i) + X(d_j) = X(d_k), \forall d_i, d_j, d_k \in D.$$

To define a ratio scale, we assume there exists a magnitude comparison operation (\oslash) that can be applied to objects in the domain (e.g., the *ratio* of the brightness of this patch to the the brightness of that patch). We then say that a scale is ratio if

$$d_i \oslash d_j \sim d_k \iff X(d_i)/X(d_j) = X(d_k), \forall d_i, d_j, d_k \in D.$$

Axiomatic scale theory is often invoked by practitioners of data mining and graphics, but it is not sufficient for determining scales on statistical graphics produced by chart algebra. The blend operation, for example, allows us to union values on different variables. We can require that blended variables share the same measurement level (e.g., diastolic and systolic blood pressure), but this will not always produce a meaningful scale. For example, we will have a meaningless composite scale if we attempt to blend height and weight, both presumably ratio variables.

We need a different level of detail so that we can restrict the blend operation more appropriately.

Unit Measurement

11.4.2

An alternative scale classification is based on units of measurement. Unit scales permit standardization and conversion of metrics. In particular, the International System of Units (SI) (Taylor, 1997) unifies measurement under transformation rules encapsulated in a set of base classes. These classes are *length*, *mass*, *time*, *electric current*, *temperature*, *amount of substance*, and *luminous intensity*. Within the base classes, there are default metrics (meter, kilogram, second, etc.) and methods for converting from one metric to another. From these base classes, a set of derived classes yields measurements such as *area*, *volume*, *pressure*, *energy*, *capacitance*, *density*, *power*, and *force*. Table 11.2 shows some examples of several SI base classes, derived classes, and an example of an economic base class that is not in SI. The *currency* class is time dependent, since daily exchange rates determine conversion rules and an inflation adjustment method varies with time.

Most of the measurements in the SI system fit within the interval and ratio levels of Stevens' system. There are other scales fitting Stevens' system that are not classified within the SI system. These involve units such as *category* (state, province, country, color, species), *order* (rank, index), and *measure* (probability, proportion, percent). And there are additional scales that are in neither the Stevens nor the SI system, such as *partial order*.

For our purposes, unit measurement gives us the level of detail needed to construct a numerical or categorical scale. We consider unit measurement a form of strong typing that enables reasonable default behavior. Because of the class structure and conversion methods, we can handle labels and relations for derived quantities such as miles-per-gallon, gallons-per-mile, and liters-per-kilometer. Furthermore, automatic unit conversion within base and derived classes allows

Table 11.2. Typical unit measurements

Length	Mass	Temperature	Time	Volume	Currency
meter	kilogram	kelvin	second	liter	dollar
point	gram	rankine	minute	teaspoon	euro
pica	grain	celsius	hour	tablespoon	pound
inch	slug	fahrenheit	day	cup	yen
foot	carat		week	pint	rupee
yard			month	quart	dinar
mile			quarter	gallon	
furlong			year	bushel	
fathom			century	barrel	

meaningful blends. As with domain check overrides in a database (Date, 1990), we allow explicit type overrides for the blend operation.

11.4.3 Transformations

We frequently compute transformations of variables in constructing graphics. Sometimes, we employ statistical transformations to achieve normality so that we can apply classical statistical methods such as linear regression. Other times, we transform to reveal local detail in a graphic. It helps to apply a log transform, for example, to stretch out small data values in a display. We might do this even when not applying statistical models.

These types of transformations fall within the scale stage of the grammar of graphics system. Because GOG encapsulates variable transformations within this stage, it accomplishes two tasks at the same time: 1) the values of the variables are transformed prior to analysis and display, and 2) nice scale values for axes and legends are computed based on the transformation. Figure 11.7 shows an example of this process for the city data. In order to highlight population changes in small cities, we represent the populations on a log scale. The algebraic expression is the same as in Fig. 11.5: $city * (pop1980 + pop2000)$. Now we see that most of the cities gained population between 1980 and 2000 but half the US namesakes lost population.

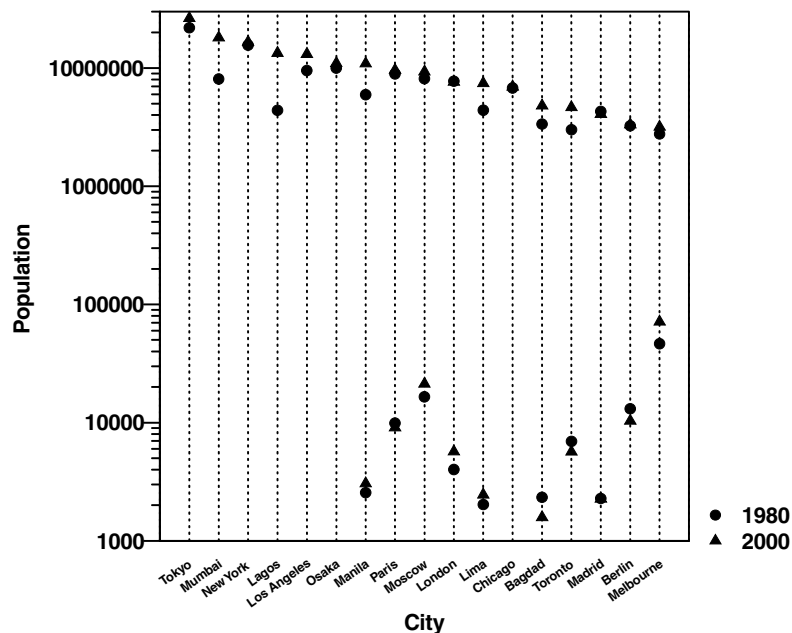


Figure 11.7. $city * (pop1980 + pop2000)$, $ylog$

Statistics

Visualization and statistics are inseparable. Statisticians have known this for a long time, but non-statisticians in the visualization field have largely ignored the role of statistics in charts, maps, and graphics. Non-statisticians often believe that visualization follows data analysis. We aggregate, summarize, model, and *then* display the results. In this view, visualization is the last step in the chain and statistics is the first.

In GOG, statistics falls in the middle of the chain. The consequence of this architecture is that statistical methods are an integral part of the system. We can construct dynamic graphics, in which statistical methods can be changed (for exploratory purposes) without altering any other part of the specification and without restructuring the data. By including statistical methods in its architecture, GOG also makes plain the independence of statistical methods and geometric displays. There is no necessary connection between regression methods and curves or between confidence intervals and error bars or between histogram binning and histograms.

In GOG, the statistics component receives a varset, computes various statistics, and outputs another varset. In the simplest case, the statistical method is an identity. We do this for scatterplots. Data points are input and the same data points are output. In other cases, such as histogram binning, a varset with n rows is input and a varset with k rows is output, where k is the number of bins ($k < n$). With smoothers (regression or interpolation), a varset with n rows is input and a varset with k rows is output, where k is the number of knots in a mesh over which smoothed values are computed. With point summaries (means, medians, ...), a varset with n rows is input and a varset with one row is output. With regions (confidence intervals, ranges, ...), a varset with n rows is input and a varset with two rows is output.

Understanding how the statistics component works reveals an important reason for mapping values to cases in a varset rather than the other way around. If

$$A = [\mathbb{R}, \{\langle \cdot \rangle, \langle \cdot, \cdot \rangle, \dots\}, \{1.5 \rightarrow \langle 1 \rangle, 2.7 \rightarrow \langle 2 \rangle, 1.8 \rightarrow \langle 3 \rangle\}] ,$$

then

$$\text{mean}(A) = [\mathbb{R}, \{\langle \cdot \rangle, \langle \cdot, \cdot \rangle, \dots\}, \{2.0 \rightarrow \langle 1, 2, 3 \rangle\}] .$$

Notice that the list of caseIDs that is produced by *mean()* is contained in the one row of the output varset. We do not lose case information in this mapping, the way we do when we compute results from an ordinary SQL query on a database or when we compute a data cube for an OLAP or when we pre-summarize data to produce a simple graphic. This aspect of GOG is important for dynamic graphics systems that allow *drill-down* or queries regarding metadata when the user hovers over a particular graphic element.

Figure 11.8 shows an application of a statistical method to the city data. We linearly regress 2000 population on 1980 population to see if population growth

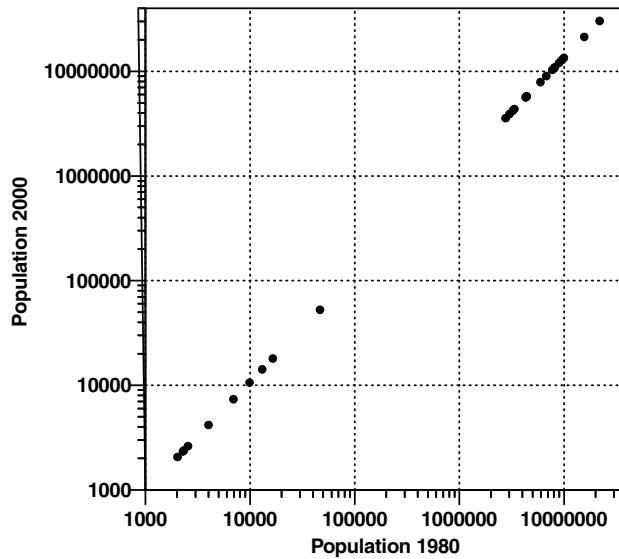


Figure 11.8. `pop1980 * estimate(pop2000), xlog, ylog`

is proportional to city size. On log-log scales, the estimated values fall on a line whose slope is greater than 1, suggesting that larger cities grow faster than smaller. Ordinarily, we would draw a line to represent the regression and we would include the data points as well. We would also note that Lagos grew at an unusual rate (with a Studentized residual of 3.4). Nevertheless, our main point is to show that the statistical regression produces data points that are exchangeable with the raw data insofar as the entire GOG system is concerned. How we choose to represent the regressed values graphically is the subject of the next section.

11.6 Geometry

GOG presumes no connection between a statistical method and a geometric representation. Histogram bins need not be represented by histograms. Tukey schematic plots (his original word for box plots) need not be represented by boxes and whiskers. Regressions need not be represented by lines or curves. Separating geometry from data (and from other graphical aspects such as coordinate systems) is what gives GOG its expressive power. We choose geometric representation objects independently of statistical methods, coordinate systems, or aesthetic attributes.

As Fig. 11.1 indicates, the geometry component of GOG receives a varset and outputs a geometric graph. A geometric graph is a subset of \mathbb{R}^n . For our purposes,

we will be concerned with geometric graphs for which $1 \leq n \leq 3$. Geometric graphs are enclosed in bounded regions:

$$B^n \subset [a_1, b_1] \times \dots \times [a_n, b_n]$$

These intervals define the edges of a bounding box or region in n -dimensional space. There are two reasons we need bounded regions. First, in order to define certain useful geometric graphs, we need concepts like the *end* of a line or the *edge* of a rectangle. Second, we want to save ink and electricity. We don't want to take forever to compute and draw a line.

Geometric graphs are produced by graphing functions $F : B^n \rightarrow \mathbb{R}^n$ that have geometric names like *line()* or *tile()*. A geometric graph is the image of F . And a graphic, as used in the title of this chapter, is the image of a graph under one or more aesthetic functions. Geometric graphs are not visible. As Bertin (1967) points out, visible elements have features not present in their geometric counterparts.

Figures 11.9 and 11.10 illustrate the exchangeability of geometry and statistical methods. The graphics are based on UN data involving 1990 estimates of female life expectancy and birth rates for selected world countries. Figure 11.9 shows four different geometric graphs – *point*, *line*, *area*, and *bar* – used to represent a confidence interval on a linear regression. Figure 11.10 shows one geometric graph used to represent four different statistical methods – local mean, local range, quadratic regression, and linear regression confidence interval.

This exchangeability produces a rich set of graphic forms with a relatively small number of geometric graphs. Table 11.3 contains these graphing methods. The *point()* graphing function produces a geometric point, which is an n -tuple. This function can also produce a finite set of points, called a *multipoint* or a *point cloud*. The set of points produced by *point()* is called a *point graph*.

The *line()* graphing function is a bit more complicated. Let B^m be a bounded region in \mathbb{R}^m . Consider the function $F : B^m \rightarrow \mathbb{R}^n$, where $n = m + 1$, with the following additional properties:

1. The image of F is bounded, and
2. $F(x) = (\mathbf{v}, f(\mathbf{v}))$, where $f : B^m \rightarrow \mathbb{R}$ and $\mathbf{v} = (x_1, \dots, x_m) \in B^m$.

If $m = 1$, this function maps an interval to a functional curve on a bounded plane. And if $m = 2$, it maps a bounded region to a functional surface in a bounded 3D space. The *line()* graphing function produces these graphs. Like *point()*, *line()* can produce a finite set of lines. A set of lines is called a *multiline*. We need this capability for representing multimodal smoothers, confidence intervals on regression lines, and other multifunctional lines.

The *area()* graphing function produces a graph containing all points within the region under the *line* graph. The *bar()* graphing function produces a set of closed intervals. An interval has two ends. Ordinarily, however, bars are used to denote a single value through the location of one end. The other end is anchored

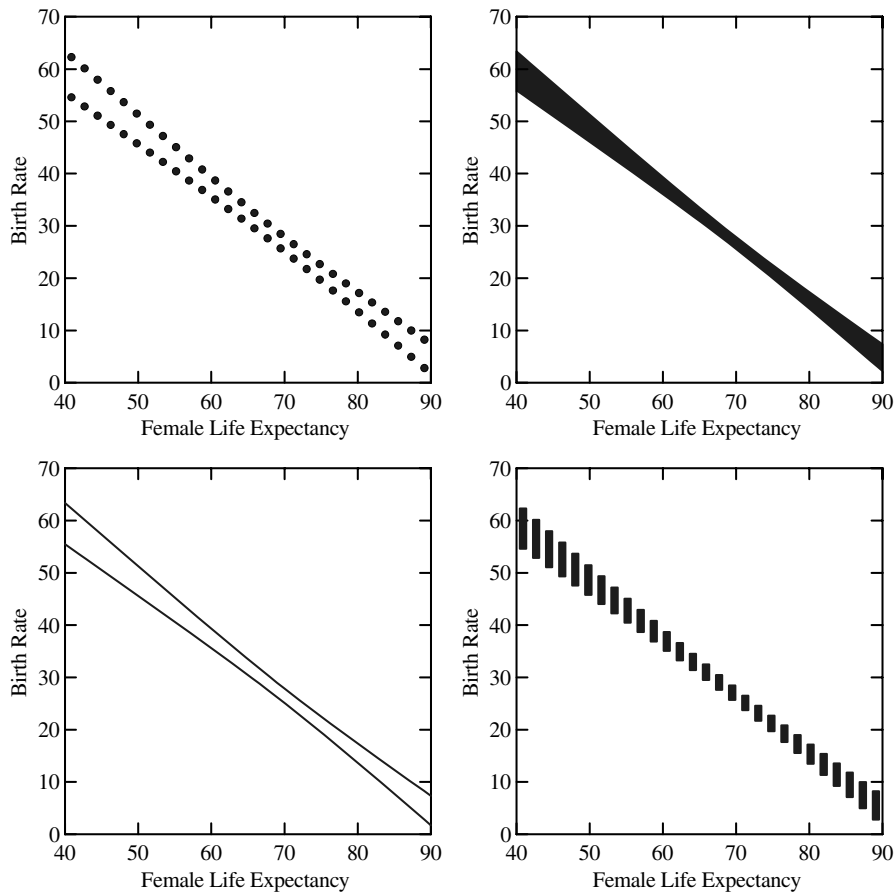


Figure 11.9. Different graph types, same statistical method

Table 11.3. Geometric Graphs

Relations	Summaries	Partitions	Networks
<i>point</i>	<i>schema</i>	<i>tile</i>	<i>path</i>
<i>line (surface)</i>		<i>contour</i>	<i>link</i>
<i>area (volume)</i>			
<i>bar (interval)</i>			
<i>histobar</i>			

at a common reference point (usually zero). The *histobar()* graphing function produces a histogram element. This element behaves like a bar except a value maps to the area of a histobar rather than to its extent. Also, histobars are glued to each other. They cover an interval or region, unlike bars.

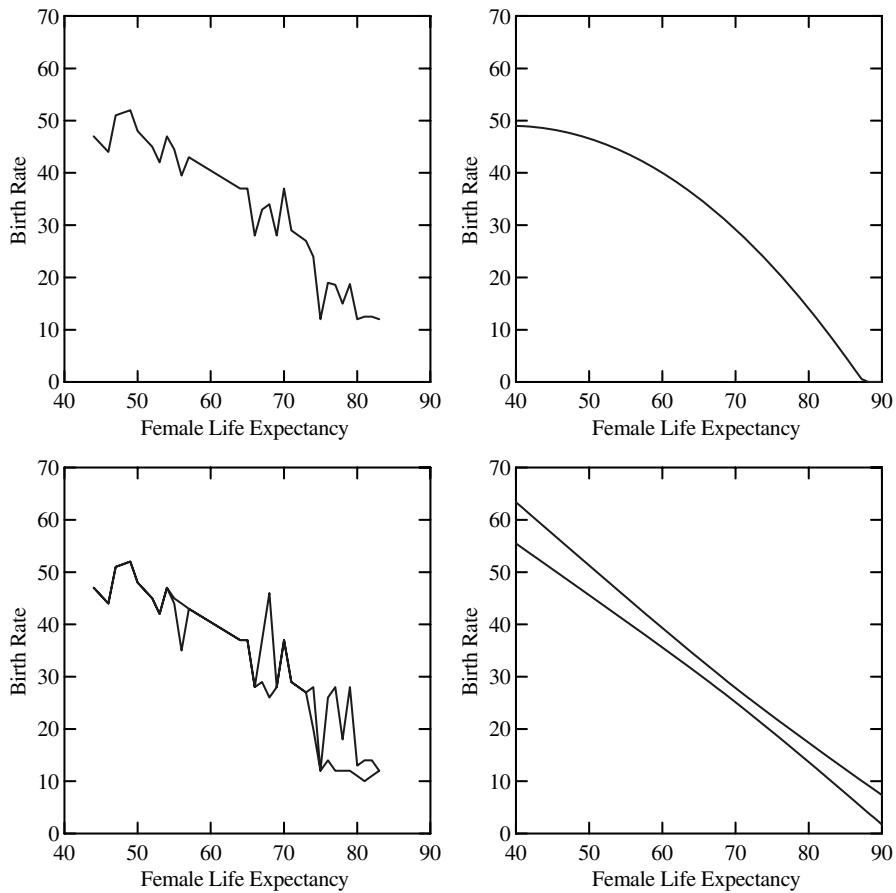


Figure 11.10. Different statistical methods, same graph type

A *schema* is a diagram that includes both general and particular features in order to represent a distribution. We have taken this usage from Tukey (1977), who invented the schematic plot, which has come to be known as the box plot because of its physical appearance. The *schema()* graphing function produces a collection of one or more points and intervals.

The *tile()* graphing function tiles a surface or space. A *tile* graph covers and partitions the bounded region defined by a frame; there can be no gaps or overlaps between tiles. The Latinate *tessellation* (for tiling) is often used to describe the appearance of the tile graphic.

A *contour()* graphing function produces contours, or level curves. A *contour* graph is used frequently in weather and topographic maps. Contours can be used to delineate any continuous surface.

The *network()* graphing function joins points with line segments (edges). Networks are representations that resemble the edges in diagrams of theoretic graphs.

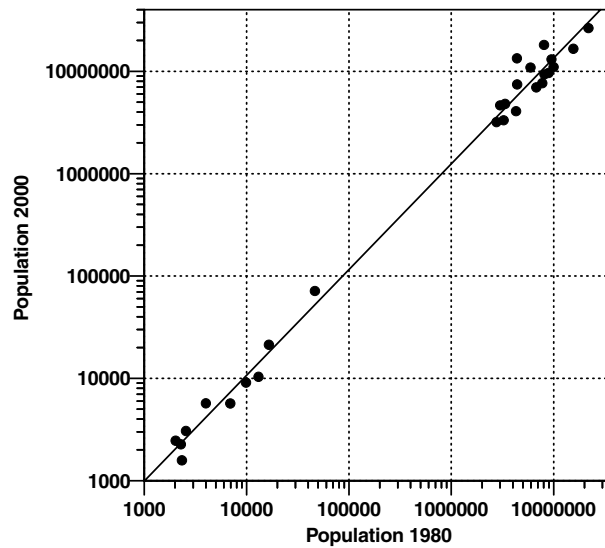


Figure 11.11. `pop1980 * {pop2000, estimate(pop2000)}, xlog, ylog, {point, line}`

Although networks join points, a point graph is not needed in a frame in order for a network graphic to be visible.

Finally, the `path()` graphing function produces a *path* that connects points such that each point touches no more than two line segments. Thus, a path visits every point in a collection of points only once. If a path is closed (every point touches two line segments), we call it a circuit. Paths often look like lines. There are several important differences between the two, however. First, lines are functional; there can be only one point on a line for any value in the domain. Paths may loop, zigzag, and even cross themselves inside a frame. Second, paths consist of segments that correspond to edges, or links between nodes. This means that a variable may be used to determine an attribute of every segment of a path.

Figure 11.11 contains two geometric objects for representing the regression we computed in Fig. 11.8. We use a *point* for representing the data and a *line* for representing the regression line.

11.7 Coordinates

The most popular types of charts employ Cartesian coordinates. The same real tuples in the graphs underlying these graphics can be embedded in many other coordinate systems, however. There are many reasons for displaying graphics in different coordinate systems. One reason is to simplify. For example, coordinate transformations can change some curvilinear graphics to linear. Another reason is to reshape graphics so that important variation or covariation is more salient

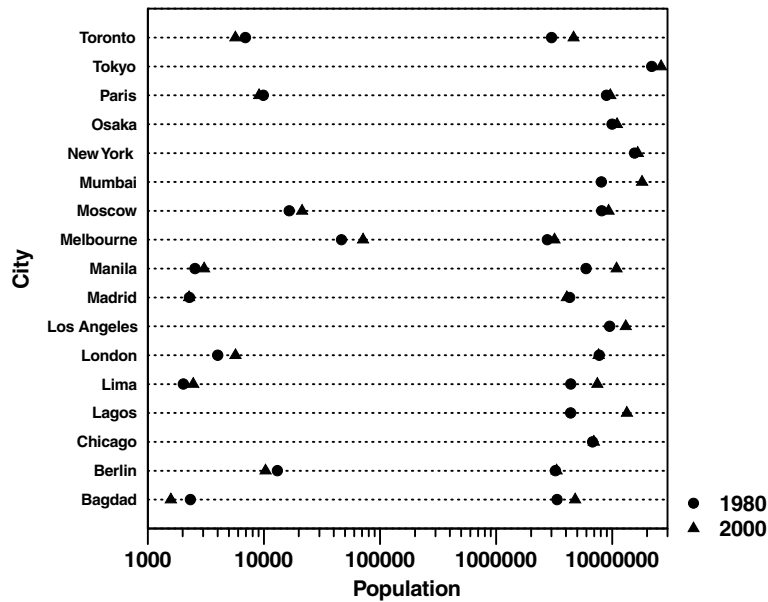


Figure 11.12. $\text{transpose}(\text{city} * (\text{pop1980} + \text{pop2000})), y\log$

or accurately perceived. For example, a pie chart is generally better for judging proportions of wholes than is a bar chart (Simkin and Hastie, 1987). Yet another reason is to match the form of a graphic to theory or reality. For example, we might map a variable to the left-closed and right-open interval $[0, 1)$ on a line or to the interval $[0, 2\pi)$ on the circumference of a circle. If our variable measures defects within a track of a computer disk drive in terms of rotational angle, it is usually better to stay within the domain of a circle for our graphic. Another reason is to make detail visible. For example, we may have a cloud with many points in a local region. Viewing those points may be facilitated by zooming in (enlarging a region of the graphic) or smoothly distorting the local area so that the points are more separated in the local region.

Wilkinson (1999) contains many examples of ordinary charts rendered in different coordinate systems. A simple example suffices for the data in this chapter. Figure 11.12 shows a transposed version of Fig. 11.7. The result of this coordinate transformation (a rotation composed with a reflection) is to make the city names more readable.

Aesthetics

The term aesthetics derives from a Greek word that means perception. The derivative modern meanings of beauty, taste, and artistic criteria arose in the 18th cen-

ture. We have chosen the name *aesthetics* to describe the class of functions that turn theoretical graphs into perceivable graphics because of its original connotations and because the modern word *perception* is subjective rather than objective; perception refers to the perceiver rather than the object. Aesthetics turn graphs into graphics so that they are perceivable, but they are not the perceptions themselves. A modern psychologist would most likely call aesthetics in this sense stimuli, aspects, or features, but these words are less germane to our purpose.

Table 11.4 summarizes these aesthetic attributes. We have grouped these attributes in five categories: *form*, *surface*, *motion*, *sound*, and *text*. This is not intended to be an exhaustive list; other attributes, such as *odor*, can be devised. The *color* aesthetic has three components: *hue*, *brightness*, and *saturation* (other color components are possible). The *texture* aesthetic includes components of *pattern*, *granularity*, and *orientation*.

Seven of these attributes are derived from the *visual variables* of Bertin (1967): *position* (position), *size* (taille), *shape* (forme), *orientation* (orientation), *brightness* (valeur), *color* (couleur), and *granularity* (grain). Bertin's *grain* is often translated as *texture*, but he really means granularity (as in the granularity of a photograph). Granularity in this sense is also related to the spatial frequency of a texture.

These aesthetic attributes do not represent the aspects of perception investigated by psychologists. This lack of fit often underlies the difficulty graphic designers and computer specialists have in understanding psychological research relevant to graphics and the corresponding difficulty psychologists have with questions asked by designers. Furthermore, these attributes are not ones customarily used in computer graphics to create realistic scenes. They are not even sufficient for a semblance of realism. Notice, for example, that *pattern*, *granularity*, and *orientation* are not sufficient for representing most of the textures needed for representing real objects. Instead, these attributes are chosen in a tradeoff between the psychological dimensions they elicit and the types of routines that can be implemented in a rendering system. Specifically,

- An attribute must be capable of representing both continuous and categorical variables.
- When representing a continuous variable, an attribute must vary primarily on one psychophysical dimension. In order to use multidimensional attributes such as color, we must scale them on a single dimension such as hue or bright-

Table 11.4. Aesthetics

Form	Surface	Motion	Sound	Text
<i>position</i>	<i>color</i>	<i>direction</i>	<i>tone</i>	<i>label</i>
<i>size</i>	<i>texture</i>	<i>speed</i>	<i>volume</i>	
<i>shape</i>	<i>blur</i>	<i>acceleration</i>	<i>rhythm</i>	
<i>rotation</i>	<i>transparency</i>		<i>voice</i>	

ness, or compute linear or nonlinear combinations of these components to create a unidimensional scale.

- An attribute does not imply a linear perceptual scale. In fact, few aesthetic attributes scale linearly. Some attributes such as hue scale along curvilinear segments in two- or three-dimensional space. All linear scales are unidimensional but not all unidimensional scales are linear.
- A perceiver must be able to report a value of a variable relatively accurately and effortlessly when observing an instance of that attribute representing that variable.
- A perceiver must be able to report values on each of two variables relatively accurately upon observing a graphic instantiating two attributes. This task usually, but not necessarily, requires selective attention. This criterion probably isn't achievable for all of our attributes and may not even be achievable for any pair of them. But any attribute that is clearly non-separable with another should be rejected for our system. It is too much to expect, of course, that higher order interactions among attributes be non-existent. Much of the skill in graphic design is knowing what combinations of attributes to avoid.
- Each attribute must name a distinct feature in a rendering system. We cannot implement an attribute that does not uniquely refer to a drawable (or otherwise perceivable) feature. An attribute cannot be mapped to a miscellaneous collection of widgets or controls, for example.

We have attempted to classify aesthetics so that they are orthogonal in a design sense. One must not assume that this implies they are uncorrelated in our perceptual mechanisms, however. Orthogonalization in design means making every dimension of variation that is available to one object available to another. How these variations are perceived is another matter. Many aesthetic attributes, even ones such as *size* or *position* that are usually considered visual, need not be perceived visually. There is nothing in the definition of a graphic to limit it to vision. Provided we use devices other than computer screens and printers, we can develop graphical environments for non-sighted people or for those unable to attend to a visual channel because, perhaps, they are busy, restrained, or multiprocessing. Touch, hearing, even smell can be used to convey information with as much detail and sensitivity as can vision.

Every one of the figures in this chapter incorporates several aesthetics. Without aesthetic functions, they would not be visible. Consequently, we will not add a figure to illustrate other aesthetics, particularly since we are constrained in publishing format. Note, however, that in addition to using the *position* aesthetic function in every graphic, we have employed *shape* to differentiate symbols. Note, also, that *position* aesthetics are usually referenced by axes and *shape* and other aesthetics are usually referenced by legends.

Our discussion of the seven primary GOG components ends here. But there are several important topics remaining. We will first examine issues in graphics layout, and then conclude with a discussion of the relation between graphics algebra and statistical design models.

11.9 Layout

Chart algebra does not determine the physical appearance of charts plotted on a screen or paper. It simply produces a set of tuples (x_1, \dots, x_p) that can be rendered using geometric primitives and a layout interpreter. If we have 2-tuples, then we can render them directly on a computer screen or piece of paper. If we have 3-tuples, then we can use a perspective projection to render them on the plane. Higher-order tuples require a layout scheme to embed all dimensions in the plane. Various layout schemes are attempts to solve a graphic representation problem: how to transform a p -dimensional vector space to a 2-dimensional space so that we can perceive structures in the higher dimensional space by examining the 2-dimensional space. We will discuss several approaches in this section.

Projection

One scheme is to employ a linear or nonlinear projection from p -dimensions to two. This may cause loss of information because a projection onto a subspace is many-to-one. Also, projection is most suitable for displaying points or $\{V, E\}$ graphs. It is less suitable for many geometric chart types such as bars and pies. Nevertheless, some 2D projections have been designed to capture structures contained in subspaces, such as manifolds, simplices, or clusters (Friedman, 1987). Other popular projection methods are principal components and multidimensional scaling (Hastie et al., 2001).

Sets of Functions

A second possibility is to map a set of n points in \mathbb{R}^p one-to-one to a set of n functions in \mathbb{R}^2 . A particularly useful class of functions is formed by taking the first p terms in a Fourier series as coefficients for (x_1, \dots, x_p) (Andrews, 1972). Another useful class is the set of Chebyshev orthogonal polynomials. A popular class is the set of $p - 1$ piecewise linear functions with (x_1, \dots, x_p) as knots, often called *parallel coordinates* (Inselberg, 1984; Wegman, 1985).

An advantage of function space representations is that there is no loss of information, since the set of all possible functions for each of these types in \mathbb{R}^2 is infinite. Orthogonal functions (such as Fourier and Chebyshev) are useful because zero inner products are evidence of linear independence. Parallel coordinates are useful because it is relatively easy to decode values on particular variables. A disadvantage of functional representations is that manifolds, solids, and distances are difficult to discern.

Recursive Partitioning

A third possibility is recursive partitioning. We choose an interval $[u_1, u_2]$ and partition the first dimension of \mathbb{R}^p into a set of connected intervals of size $(u_2 - u_1)$, in the same manner as histogram binning. This yields a set of rectangular subspaces of \mathbb{R}^p . We then partition the second dimension of \mathbb{R}^p similarly. This second partition produces a set of rectangular subspaces within each of the previous subspaces. We

continue choosing intervals and partitioning until we finish the last dimension. We then plot each subspace in an ordering that follows the ancestry of the partitioning. Recursive partitioning layout schemes have appeared in many guises: $\mathbb{R}^p \mapsto \mathbb{R}^3$ (Feiner and Beshers, 1990), $\mathbb{R}^p \mapsto \mathbb{R}^2$ (Mihalisin et al., 1991), $\mathbb{R}^4 \mapsto \mathbb{R}^2$ (Becker et al., 1996).

There are several modifications we may make to this scheme. First, if a dimension is based on a categorical variable, then we assume $(u_2 - u_1) = 1$, which assures one partition per category. Second, we need not partition a dimension into equal intervals; instead, we can make $[u_1, u_2]$ adaptive to the density of the data (Wilkinson, 1999, page 186). Third, we can choose a variety of layouts for displaying the nodes of the partitioning tree. We can display the cells as an n -ary tree, which is the method used by popular decision-tree programs. Or, we can alternate odd/even dimensions by plotting horizontally/vertically. This display produces a 2D nested table, which has been variously named a *mosaic* (Hartigan and Kleiner, 1981) or *treemap* (Johnson and Schneiderman, 1991). We use this latter scheme for the figures in this article.

This rectangular partitioning resembles a 2D rectangular fractal generator. Like simple projection, this method can cause loss of information because aggregation occurs within cells. Nevertheless, it yields an interpretable 2D plot that is familiar to readers of tables.

Because recursive partitioning works with either continuous or categorical variables, there is no display distinction between a table and a chart. This equivalence between tables and graphs has been noted in other contexts (Shoshani, 1997; Pedersen et al., 2002). With recursive partitioning, we can display tables of charts and charts of tables.

Analytics

11.10

If conclusions based on statistical charts are to be useful, we must identify and interpret the statistical models underlying charts. A statistical model determines how the location of a representation element (point, line, ...) in a frame (a measurable region of representation) is computed from the values of a variable. Statistical models usually (but not necessarily) incorporate error terms and help us to articulate the domains of generalizations and inferences we make from examining a chart. Glymour et al. (1996) summarize these issues from a data mining context. Because chart algebra is based on statistical design algebras, it can be used to generate statistical models for visual data mining or predictive analytics.

This section presents the statistical model equivalents of chart algebra expressions. In each subsection, we show the chart algebra notation on the left of each equivalence expression and the statistical model notation on the right. The terms on the left comprise varsets and the terms on the right comprise variables. Note that some symbols (e.g., +) are common to both notations but have different meanings. The general linear statistical models presented in this section are due to

(Fisher, 1925; Fisher, 1935). More recent introductions to the design notation used for statistical models are (Heiberger, 1989) and (Kutner et al., 1996).

11.10.1 Statistical Model Equivalents

In the following subsections, we assume a functional model $Z = f(X, Y)$, where Z is a (possibly multivariate) variable. Z corresponds to a varset Z , which itself might be produced from a chart algebra expression. In statistical terms, we sometimes call Z a *dependent* variable and X and Y *independent* variables. In this section, we ignore Z and focus on expressions involving X and Y . These expressions are used to construct statistical models that help to predict or estimate Z .

Cross

$$X * Y \sim \mathcal{C} + X + Y + XY$$

The cross operator corresponds to a fully factorial experimental design specification. This design employs a product set that includes every combination of levels of a set of experimental factors or treatments. The terms on the right of the similarity represent the linear model for fitting fully factorial designs. The terms in the model are:

\mathcal{C} : constant term (grand mean)

X : levels of factor X (X main effect)

Y : levels of factor Y (Y main effect)

XY : product of factors X and Y(interactions)

We could use boldface for the variables on the right because the most general form of the model includes factors (multidimensional categorical variables) having more than one level. These multivariate terms consist of sets of binary categorical variables whose values denote presence or absence of each level in the factor. Alternatively, terms based on continuous variables are called *covariates*.

An example of a two-way factorial design would be the basis for a study of how teaching method and class size affect the job satisfaction of teachers. In such a design, each teaching method (factor X) is paired with each class size (factor Y) and teachers and students in a school are randomly assigned to the combinations.

Nest

$$X/Y \sim \mathcal{C} + Y + X(Y)$$

The terms on the right of the similarity are:

C : constant term

Y : levels of factor Y (Y main effect)

$X(Y)$: X levels nested within levels of Y

The term $X(Y)$ represents the series $X | (Y = Y_1) + X | (Y = Y_2) + \dots$

Notice that there is no interaction term involving X and Y because X is nested within Y . Not all combinations of the levels of X and Y are defined. An example of a nested design would be the basis for a study of the effectiveness of different teachers and schools in raising reading scores. Teachers are nested within schools when no teacher in the study can teach at more than one school. With nesting, two teachers with the same name in different schools are different people. With crossing, two teachers with the same name in different schools may be the same person.

Blend

$$X + Y \sim C + F_{XY}$$

The terms on the right of the similarity are:

C : constant term

F_{XY} : function of X and Y (e.g., $X - Y$)

The blend operator usually corresponds to a time series design. In such a design, we predict using functions of a time series. When the blend involves dependent variables, this is often called a repeated measures design. The simplest case is a prediction based on first differences of a series. Time is not the only possible dimension for ordering variables, of course. Other multivariate functional models can be used to analyze the results of blends (Ramsay and Silverman, 1997).

An example of a repeated measures design would be the basis for a study of improvement in reading scores under different curricula. Students are randomly assigned to curriculum and the comparison of interest involves differences between pre-test and post-test reading scores.

It would appear that analytics have little to do with the process of building a chart. If visualization is at the end of a data-flow pipeline, then statistics is simply a form of pre-processing. In our model, however, analytics are an intrinsic part of chart construction. Through chart algebra, the structure of a graph implies a statistical model. Given this model, we can employ likelihood, information, or goodness-of-fit measures to identify parsimonious models. We will explore some graphic uses of statistical models in this section.

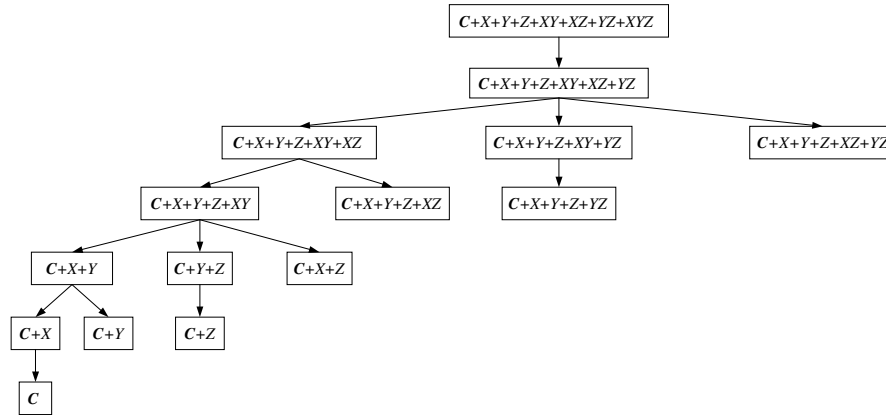


Figure 11.13. Model subset tree

11.10.2 Subset Model Fitting

The factorial structure of most chart algebra expressions can produce rather complex models. We need to consider strategies for selecting subset models that are adequate fits to the data. We will discuss one simple approach in this section. This approach involves eliminating interactions (products of factors) in factorial designs.

Interactions are often regarded as nuisances because they are difficult to interpret. Comprehending interactions requires thinking about partial derivatives. A three-way interaction XYZ , for example, means that the relation between X and Y depends on the level of Z . And the relation between X and Z depends on the level of Y . And the relation between Y and Z depends on the level of X . Without any interaction, we can speak about these simple relations unconditionally. Thus, one strategy for fitting useful subset models is to search for subsets with as few interactions as possible. In this search, we require that any variables in an interaction be present as a main-effect in the model.

Figure 11.13 shows a submodel tree for the three-way crossing $X * Y * Z$. Not all possible submodels are included in the tree, because the convention in modeling factorial designs is to include main effects for every variable appearing in an interaction. This reduces the search space for plausible submodels. By using branch-and-bound methods, we can reduce the search even further. Mosteller and Parunak (1985) and Linhart and Zucchini (1986) cover this area in more detail.

11.10.3 Lack of Fit

Statistical modeling and data mining focus on regularity: averages, summaries, smooths, and rules that account for the significant variation in a dataset. Often, however, the main interest of statistical graphics is in locating aspects that are

discrepant, surprising, or unusual: under-performing sales people, aberrant voting precincts, defective parts.

An *outlier* is a case whose data value and fitted value (using some model) are highly discrepant relative to the other data-fitted discrepancies in the dataset. (Barnett and Lewis, 1994). Casewise discrepancies are called *residuals* by statisticians. Outliers can be flagged in a display by highlighting (e.g., coloring) large residuals in the frame. Outliers are only one of several indicators of a poorly fit model, however. Relative badness-of-fit can occur in one or more cells of a table, for example. We can use subset modeling to highlight such cells in a display. Sarawagi et al. (1998) do this for log-linear models. Also, we can use autocorrelation and cross-correlation diagnostic methods to identify dependencies in the residuals and highlight areas in the display where this occurs.

Scalability

 11.10.4

Subset design modeling is most suited for deep and narrow (many rows, few columns) data tables or low-dimensional data cubes. Other data mining methods are designed for wide data tables or high-dimensional cubes (Hand et al., 2001; Hastie et al., 2001). Subset design modeling makes sense for visualization applications because the design space in these applications does not tend to be high-dimensional. Visual data exploration works best in a few dimensions. Higher-dimensional applications work best under the guidance of other data mining algorithms.

Estimating design models requires $O(n)$ computations with regard to cases, because only one pass through the cases is needed to compute the statistics for estimating the model. Although computing design models can be worse-case $O(p^2)$ in the number of dimensions, sparse matrix methods can be used to reduce this overhead because many of the covariance terms are usually zero.

An Example

 11.10.5

Smoothing data reveals systematic structure. Tukey (1977) used the word in a specific sense, by pairing the two equations

$$data = fit + residual$$

$$data = smooth + rough$$

Tukey's use of the word is different from other mathematical meanings, such as functions having many derivatives.

We smooth data in graphics to highlight selected patterns in order to make inferences. We present an example involving injury to the heads of dummies in government frontal crash tests. Figure 11.14 shows NHTSA crash test results for selected vehicles tested before 1999. The dependent variable shown on the horizontal axis of the chart is the Head Injury Index computed by the agency. The

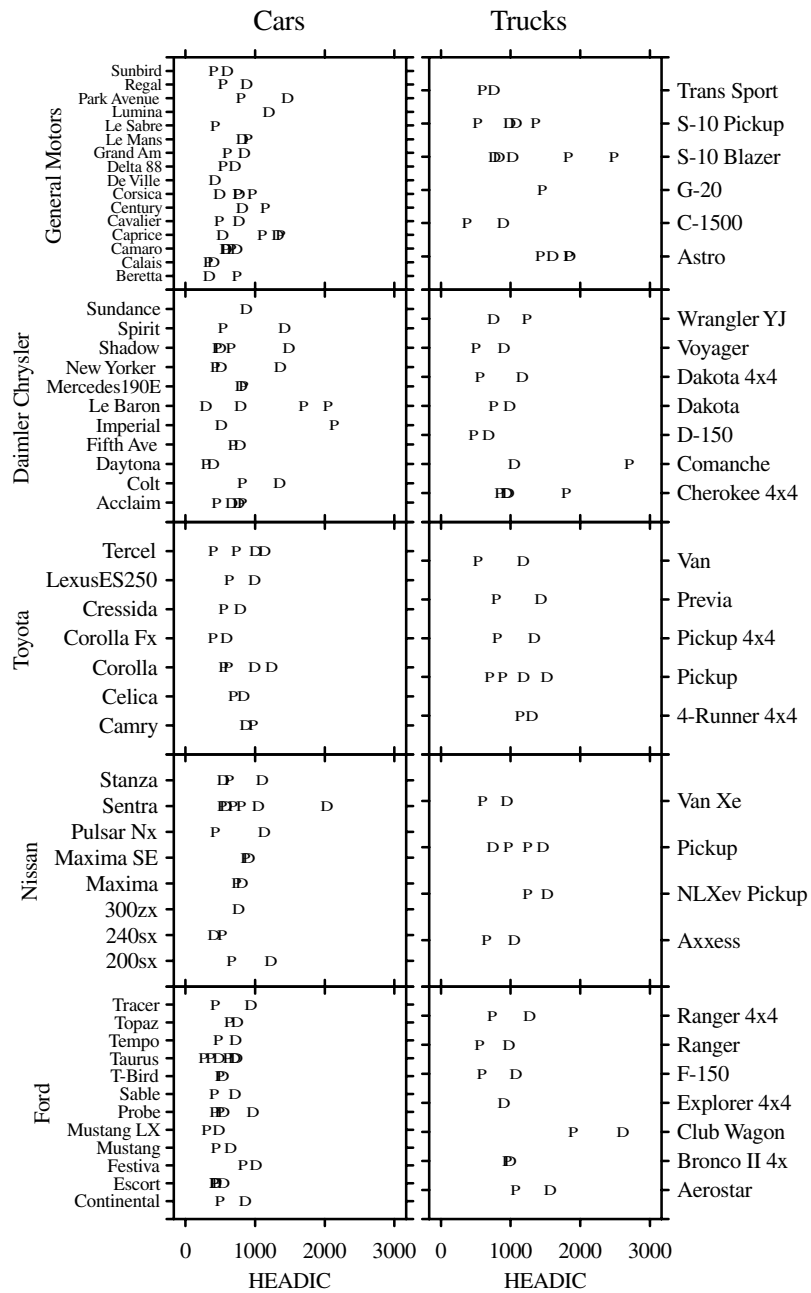


Figure 11.14. Crash data

full model is generated by the chart algebra $H * T / (M * V) * O$. This expression corresponds to the model:

$$H = \mathcal{C} + M + V + O + T(MV) + MV + MO + VO + OT(MV) + MVO$$

where the symbols are:

H : Head Injury Index

\mathcal{C} : constant term (grand mean)

M : Manufacturer

V : Vehicle (car/truck)

O : Occupant (driver/passenger)

T : Model

This display is difficult to interpret. We need to fit a model and order the display to reveal the results of the model fit. Fig. 11.15 charts fitted values from the following subset model:

$$H = \mathcal{C} + V + O + T(MV)$$

Figure 11.15 has several notable features. First, the models are sorted according to the estimated Head Injury Index. This makes it easier to compare different cells. Second, some values have been estimated for vehicles with missing values (e.g., GM G-20 driver data). Third, the trends are smoother than the raw data. This is the result of fitting a subset model. We conclude that passengers receive more head injuries than drivers, occupants of trucks and SUVs (sports utility vehicles) receive more head injuries than occupants of cars, and occupants of some models receive more injuries than occupants of others.

Software

11.11

Four systems have so far implemented the algebra described in this chapter. Wilkinson et al. (2000) developed a system called nViZn, which used the algebra to present interactive graphics in a Web environment. Norton et al. (2001) used the algebra to structure and display data in a real-time streaming environment; their system is called Dancer. Wills (2002) developed a server-based presentation graphics system with an XML schema for GOG; this system is called ViZml. Stolte et al. (2002) used the algebra to develop a visualization front-end for an OLAP; their system is called Polaris.

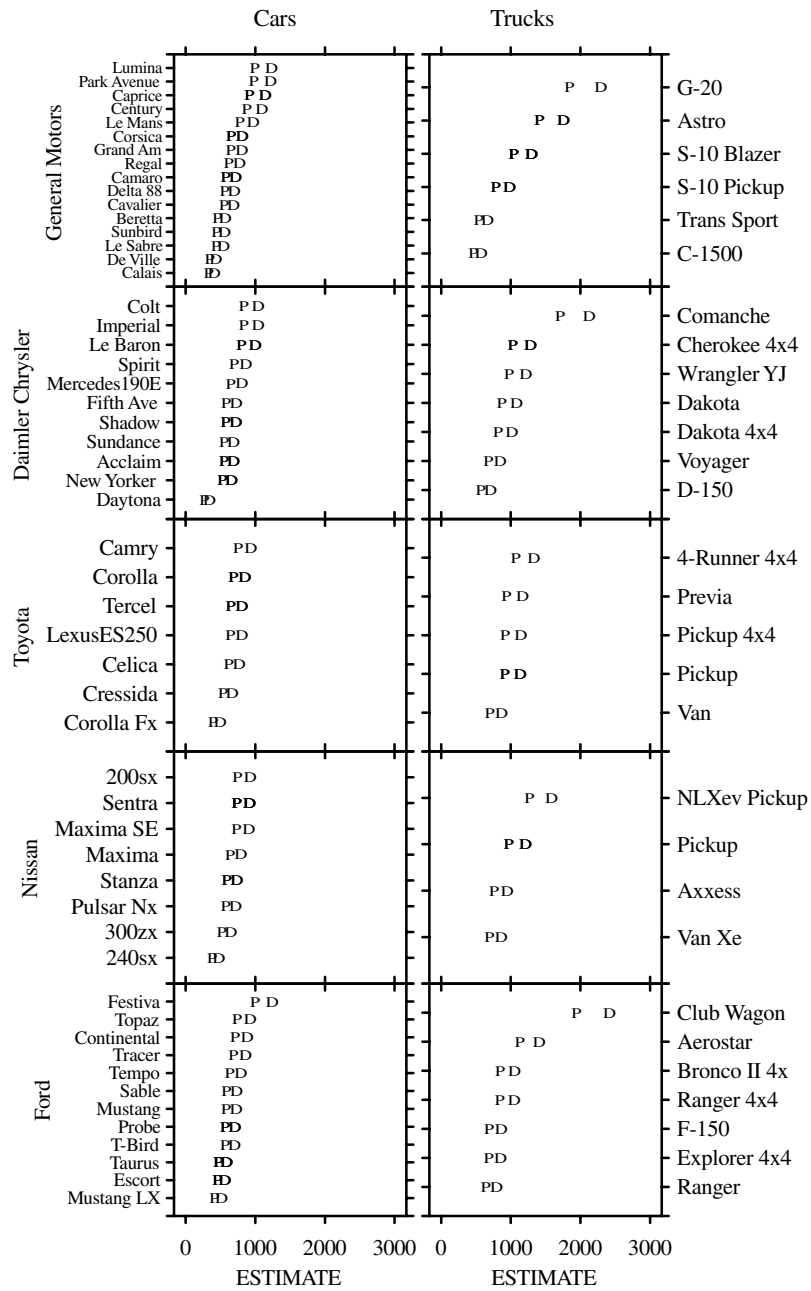


Figure 11.15. Smoothed crash data

Conclusion

11.12

Many of the pieces that motivate graphics algebra have been lying around for decades: experimental design algebras, relational algebras, table algebras. These algebras emerged from separate disciplines, so that in most instances, researchers have been unaware of developments in the other disciplines. What is new about chart algebra is the explicit and formal equivalence between the data structures needed for statistical models and the methods for displaying them in charts. In a general sense, this equivalence allows us to think about manipulating data by manipulating statistical representation elements of a chart.

The GOG project has had several purposes. One, of course, is to develop statistical graphics systems that are exceptionally powerful and flexible. Another is to understand the steps we all use when we generate charts and graphs. This understanding leads to a formalization of the problem that helps to integrate the miscellaneous set of techniques that have comprised the field of statistical graphics over several centuries. Another purpose is to develop, ultimately, intelligent systems that can 1) graph data without human specification and 2) read already published statistical graphics to recover data and interpret relationships. Finally, we hope to define problem specification and user interaction in a way that enables graphics systems to be understood by ordinary people as well as by statisticians. By formally relating display structure and statistical models, we can produce environments in which users interact with data, receive guidance, and draw conclusions that are based on appropriate statistical inferences.

References

- Abiteboul, S., Fischer, P. C. and Schek, H. J. (1989). *Nested relations and complex objects in databases*. Springer-Verlag, New York.
- Agrawal, R., Gupta, A. and Sarawagi, S. (1997). Modeling multidimensional databases. In Gray, A. and Larson, P.-Å (eds), *Proceedings of 13th International Conference of Data Engineering (ICDE)*, IEEE Computer Society, pp. 232–243.
- Andrews, D. (1972). Plots of high dimensional data. *Biometrics*, 28: 125–136.
- Barnett, V. and Lewis, T. (1994). *Outliers in Statistical Data*. John Wiley & Sons, New York.
- Becker, R. A., Cleveland, W. S. and Shyu, M.-J. (1996). The design and control of trellis display. *Journal of Computational and Statistical Graphics*, 5: 123–155.
- Bertin, J. (1967). *Smiologie Graphique*. Editions Gauthier-Villars, Paris.
- Date, C. (1990). What is a domain? In Date, C. (ed), *Relational Database Writings 1985–1989*, Addison-Wesley, Reading, MA, pp. 213–313.
- Date, C. J. and Darwen, H. (1992). Relation-valued attributes. In Date, C. J. and Darwen, H. (eds), *Relational Database: Writings 1989–1991*, Addison-Wesley, Reading, MA, pp. 75–98.

- Feiner, S. and Beshers, C. (1990). Worlds within worlds: Metaphors for exploring n-dimensional virtual worlds. In *Proceedings of ACM UIST '90*. ACM Press, pp. 76–83.
- Fisher, R. (1925). *Statistical Methods for Research Workers*. Oliver and Boyd, Edinburgh.
- Fisher, R. (1935). *The Design of Experiments*. Oliver and Boyd, Edinburgh.
- Friedman, J. (1987). Exploratory projection pursuit. *Journal of the American Statistical Association*, 82: 249–266.
- Glymour, C., Madigan, D., Pregibon, D. and Smyth, P. (1996). Statistical inference and data mining. *Communications of the ACM*, 39, 11: 35–41.
- Gyssens, M., Lakshmanan, L. V. S. and Subramanian, I. N. (1996). Tables as a paradigm for querying and restructuring (extended abstract). In *Proceedings of the fifteenth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems*, ACM Press, Montreal, Quebec, Canada, pp. 93–103.
- Hand, D. J., Mannila, H. and Smyth, P. (2001). *Principles of Data Mining: Adaptive Computation and Machine Learning*. MIT Press, Cambridge, MA.
- Hartigan, J. and Kleiner, B. (1981). Mosaics for contingency tables. In *Computer Science and Statistics: Proceedings of the 13th Symposium on the Interface*, pp. 268–273.
- Hastie, T., Tibshirani, R. and Friedman, J. H. (2001). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, New York.
- Heiberger, R. M. (1989). *Computation for the Analysis of Designed Experiments*. John Wiley & Sons, New York.
- Inselberg, A. (1984). The plane with parallel coordinates. *The Visual Computer*, 1:69–91.
- Johnson, B. and Schneiderman, B. (1991). Treemaps: A space-filling app.roach to the visualization of hierarchical information structures. In *Proceedings of the IEEE Information Visualization '91*, pp. 275–282.
- Kutner, M. H., Nachtschiem, C. J., Wasserman, W. and Neter, J. (1996). *Applied Linear Statistical Models*, Richard D. Irwin, Inc., Homewood, IL.
- Linhart, H. and Zucchini, W. (1986). *Model Selection*. John Wiley & Sons, New York.
- Mackinlay, J. (1986). Automating the design of graphical presentations of relational information. *ACM Transactions on Graphics (TOG)*, 5, 2: 110–141.
- Makinouchi, A. (1977). A consideration on normal form of not-necessarily-normalized relation in the relational model. In *Proceedings of the Third International Conference on Very Large Data Bases*.
- Mendelssohn, R. C. (1974). The bureau of labor statistic's table producing language (TPL). In *Proceedings of the 1974 annual conference*, Bureau of Labor Statistics, Washington, DC, pp. 116–122.
- Mihalisin, T., Timlin, J. and Schwegler, J. (1991). Visualizing multivariate functions, data, and distributions. *IEEE Computer Graphics and Applications*, 11, 13: 28–35.
- Mosteller, F. and Parunak, A. (1985). Identifying extreme cells in a sizable contingency table: Probabilistic and exploratory app. roaches. In Hoaglin, D. C., Mosteller, F. and Tukey, J. W. (eds), *Exploring Data Tables, Trends, and Shapes*, John Wiley, New York, pp. 189–224.

- Nelder, J. A. (1965). The analysis of randomised experiments with orthogonal block structure (Parts I and II). *Proceedings of the Royal Society of London, Series A*, 283: 147–178.
- Norton, A., Rubin, M. and Wilkinson, L. (2001). Streaming graphics. *Statistical Computing and Graphics Newsletter*, 12(1): 11–14.
- Pedersen, D., Riis, K. and Pedersen, T. B. (2002). A powerful and SQL-compatible data model and query language for OLAP. In *Proceedings of the thirteenth Australasian conference on Database technologies*, Australian Computer Society, Inc., Melbourne, Victoria, Australia, pp. 121–130.
- Ramsay, J. and Silverman, B. (1997). *Functional Data Analysis*. Springer-Verlag, New York.
- Roth, S. F., Kolojechick, J., Mattis, J., Chuah, M. C., Goldstein, J. and Juarez, O. (1994). SAGE tools: a knowledge-based environment for designing and perusing data visualizations. In *Proceedings of the CHI '94 conference companion on Human factors in computing systems*, ACM Press, Boston, Massachusetts, United States, pp. 27–28.
- Sarawagi, S., Agrawal, R. and Megiddo, N. (1998). Discovery-driven exploration of OLAP data cubes. In *Proceedings of the Sixth International Conference on Extending Database Technology (EDBT)*.
- Shoshani, A. (1997). OLAP and statistical databases: similarities and differences. In *Proceedings of the sixteenth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems*, ACM Press, Tucson, Arizona, United States, pp. 185–196.
- Simkin, D. and Hastie, R. (1987). An information processing analysis of graph perception. *Journal of the American Statistical Association*, 82: 454–465.
- Stevens, S. S. (1946). On the theory of scales of measurement. *Science*, 103: 677–680.
- Stolte, C., Tang, D. and Hanrahan, P. (2002). Polaris: A system for query, analysis, and visualization of multidimensional relational databases. *IEEE Transactions on Visualization and Computer Graphics* 8, 1: 52–65.
- Taylor, B. N. (1997). The international system of units (SI). In *Special Publication 330*, NIST, Washington, DC, pp. 171–182.
- Tukey, J. W. (1977). *Exploratory Data Analysis*. Addison-Wesley Publishing Company, Reading, MA.
- Wegman, E. J. (1985). Hyperdimensional data analysis using parallel coordinates. *Journal of the American Statistical Association*, 85: 664–675.
- Wilkinson, G. N. and Rogers, C. E. (1973). Symbolic description of factorial models for analysis of variance. *Journal of the Royal Statistical Society, Series C*, 22: 392–399.
- Wilkinson, L. (1996). A graph algebra. In *Computing Science and Statistics: Proceedings of the 28th Symposium on the Interface*, pp. 341–351.
- Wilkinson, L. (1999). *The Grammar of Graphics*. Springer-Verlag, New York.
- Wilkinson, L., Rope, D., Carr, D. and Rubin, M. (2000). The language of graphics. *Journal of Computational and Graphical Statistics*, 9: 530–543.
- Wills, G. (2002). Using the java 2 platform for advanced information visualization. Unpublished paper presented at the Annual Java One conference.

Statistical User Interfaces

II.12

Sigbert Klinke

12.1	<i>Introduction</i>	380
12.2	<i>The Golden Rules and the ISO Norm 9241</i>	381
12.3	<i>Development of Statistical User Interfaces</i>	382
	Graphical User Interfaces	383
	Toolbars.....	385
	Menus	385
	Forms and Dialog Boxes	387
	Windows	387
	Response Times	390
	Catching the User Attention	390
	Command Line Interfaces and Programming Languages	392
	Error Messages.....	394
	Help System.....	398
12.4	<i>Outlook</i>	398

Introduction

A *statistical user interface* is an interface between a human user and a statistical software package. Whenever we use a statistical software package we want to solve a specific statistical problem. But very often at first it is necessary to learn specific things about the software package.

Everyone of us knows about the “religious wars” concerning the question which statistical software package/method is the best for a certain task; see Marron (1996) and Cleveland and Loader (1996) and related internet discussions. Experienced statisticians use a bunch of different statistical software packages rather than a single one; although all of the major companies (at least the marketing departments) tell us that we only need their software package.

Why do these wars, not only concerning statistical software packages, evolve? One of the reasons is that we need time to learn about the software package besides learning the statistical methodology. And the more time we need to learn to use the software package, the more we are defending “our” software package. But if we need to spend a lot of time for learning to use a statistical software package, then the question, whether this software package really has a good user interface, arises?

The basic problem is that the development of statistical software is started by experts of statistical methodology. Since they have to have a deep inside in their special fields, most of them have a very limited view to problems of other users. We generally do not consider the sex of the users, the ethnic or cultural background, the statistical knowledge or other factors when we *create* a user interface.

Thus the important questions we have to answer when we *design* a user interface are: What does the user want to do with this software package? And how can he do it effectively?

Fortunately, during years of development of software packages, we have collected a lot of experience about human behavior and specific differences because of sex, ethnic or cultural background and so on. In the book of Shneiderman (1998) a lot of rules have been collected which should help the software developers to avoid the worst problems. But the best way for developing a user interface is a development cycle of designing, testing, redesigning, testing, redesigning, ... This will take a lot of time and money, but redesigning the basic components of a software package at late development will be much more expensive or even impossible.

In this chapter only a subset of all statistical software packages, namely Data-Desk 6.0, GGobi 0.99, R 1.7.1, SPSS 11.0 (English version), SYSTAT 10, XploRe 4.6 and Mathematica 4.3 will be used for illustrating purposes (see also the section “Web references”). It covers a wide range of different statistical software packages.

In all statistical software packages we can find errors in the user interface design. User interface design is not a exact science as statistics, but it relies heavily on the way how we perceive information and how we react to it. That includes that in every design we will make errors before we can learn to avoid them in future. Therefore a lot of theories are available, partially explanatory, partially predicting, which should help us to design user interfaces.

The Golden Rules and the ISO Norm 9241

Complex statistical tasks require more and more complex statistical programs. In consequence more complex user interfaces are needed to be developed. Software developers recognized that common rules exist in simplifying the use of software systems. Shneiderman (1998) published a summary of these rules known as the “golden rules” of user interface design:

1. *Achieve consistency*
The first rule is the one which is most often violated, especially when teams of people work together. Users expect that in similar situations the software behaves similarly and requires the same actions from the user.
2. *Use shortcuts*
Beginners need a comfortable clear structured way to accomplish their task, but power users of a software package want to do their work as quickly as possible.
3. *Give informative feedback*
Users need to have a feedback on their actions. The amount of the feedback depends on the action and the user’s experience. Frequent actions require only short answers whereas rare actions require more extended answers. Beginners need more feedback whereas power users may just need acknowledgement that the task is finished.
4. *Design closed actions*
Actions should have a clear structure with a start and a well-defined end. This holds especially for dialogs and forms.
5. *Offer error prevention and easy error handling*
Software should not support erroneous input from the user and provide default values. The user should be able to recover easily from errors. If a user can revert his actions easily then this will increase his trustworthiness in the software package.
6. *Support user control*
Users prefer to initiate actions in a software package. If a user believes that he only reacts to the system he will experience a control loss.
7. *Reduce memorization*
Humans can only remember seven plus minus two information bits in their short term memory Miller, 1956. Extensive memorization to handle a software package should be avoided.

A formalization of the rules can be found, partially in very detailed instruction, in the ISO (International Standardization Organization) norm 9241. The norm itself, which distinguishes between requirements and recommendations, consists of 17 parts:

1. General introduction
2. Guidance on task requirements
3. Visual display requirements

4. Keyboard requirements
5. Workstation layout and postural requirements
6. Environmental requirements
7. Display requirements with reflections
8. Requirements for displayed colors
9. Requirements for non-keyboard input devices
10. Dialogue principles
11. Usability statements
12. Presentation of information
13. User guidance
14. Menu dialogs
15. Command dialogs
16. Direct manipulation dialogs
17. Form-filling dialogs

12.3

Development of Statistical User Interfaces

In the past we have seen the development of software according to new concepts in computer science. From the end of the 1960s/beginning of the 1970s when computers became available till now, we can distinguish several phases. In the beginning we had non-interactive, batch oriented software packages, e.g. SAS and SPSS. The idea of incremental programming and interaction lead to systems like PRIM-9 (Tukey et al., 1973, 1974) or programming languages like BASIC. Another paradigm was that the notation used should be compact, like in languages as in APL or C. The availability of personal computers with window systems introduced graphical user interface (GUI) in contrast to command line interfaces (CLI) also to statistical software packages. As mentioned in the interview with J.E. Gentle (Härdle, 2004) statistical software packages nowadays should come with both interfaces. During the last fifteen years we saw that team programming, reusability of software, network/internet computing and access to databases had their impact on programming languages (ADA, C++, Java, SQL) as well as on statistical software packages like S/S-Plus, R, XploRe, Jasp, etc.

Before we start to develop a statistical software package and a user interface (GUI or CLI), we should think about the kind of problems a user may have:

1. A user could formulate an inadequate goal, e.g. using Excel for semi-parametric modeling.
2. A user could not find the right tool/method, since the developer uses inappropriate labels, e.g. the command `paf` in XploRe should better be named `selectrows`.
3. A user could not be able to find or execute a specific method, e.g. in a statistical programming language with a lot of commands and macros, he could loose

the overview. For example, languages like XploRe or R consist of a lot of commands, macros and packages.

4. The feedback from the software package to a user action could be inappropriate or misleading, e.g. the error message `syntax error`.

The first problem can not be solved with a better interface design, but so can the latter three (Franzke, 1995). We have two ways to solve them: either we make a better user interface or the user has to spend a lot of time for learning about the interface. One of the most time consuming design error is a subtle inconsistency, for example if the parameter orders for nearly identical actions, either in formulas for GUIs or commands in CLIs, are different. The user will always lose time to look up these subtle differences.

Obviously we can not develop one user interface for all users. The slogan *Know your user* (Hansen, 1971) in detail (statistical background knowledge, cultural background, etc.) is an important building block to the success of a (statistical) software package. We can distinguish three types of users: *novice users* who need to execute a small set of simple exercises and need an informative feedback from the software package. *Periodic users* who need support for forgotten execution sequences and need to know how to extend the current software package. But they usually need only a short feedback to an executed task. A *power user* is mainly interested in fast answers and needs only very limited feedback. Some statistical software packages, XploRe and R offer even multiple GUIs for different types of users.

However, basic guidelines for all user types are:

1. consistency in the appearance
2. effective information control by the user
3. minimal memorization and minimal data entry by the user
4. flexible adaption of the data display
5. compatibility between data entry and output

Graphical User Interfaces

12.3.1

For novice users it is clear that they prefer software with GUIs (see Temple, Barker and Sloane, Inc., Temple, Barker and Sloane, Inc., 1990), but for power users this is not quite clear, see Ulich et al. (1991). There are some general drawbacks of GUIs:

1. They need valuable screen space for menus, icons, windows etc.
2. There is no consistent set of menus, icons and windows. We have to relearn them for each software package.

A look at Fig. 12.1 shows the entry screens of different statistical software packages. Here we find all elements forming a modern GUI: menu bar, toolbar(s) and multiple windows. Note that a *statistical* user interface is more than a GUI: design of the programming language, help system, basically every aspect in the interface between a user and a statistical software package.

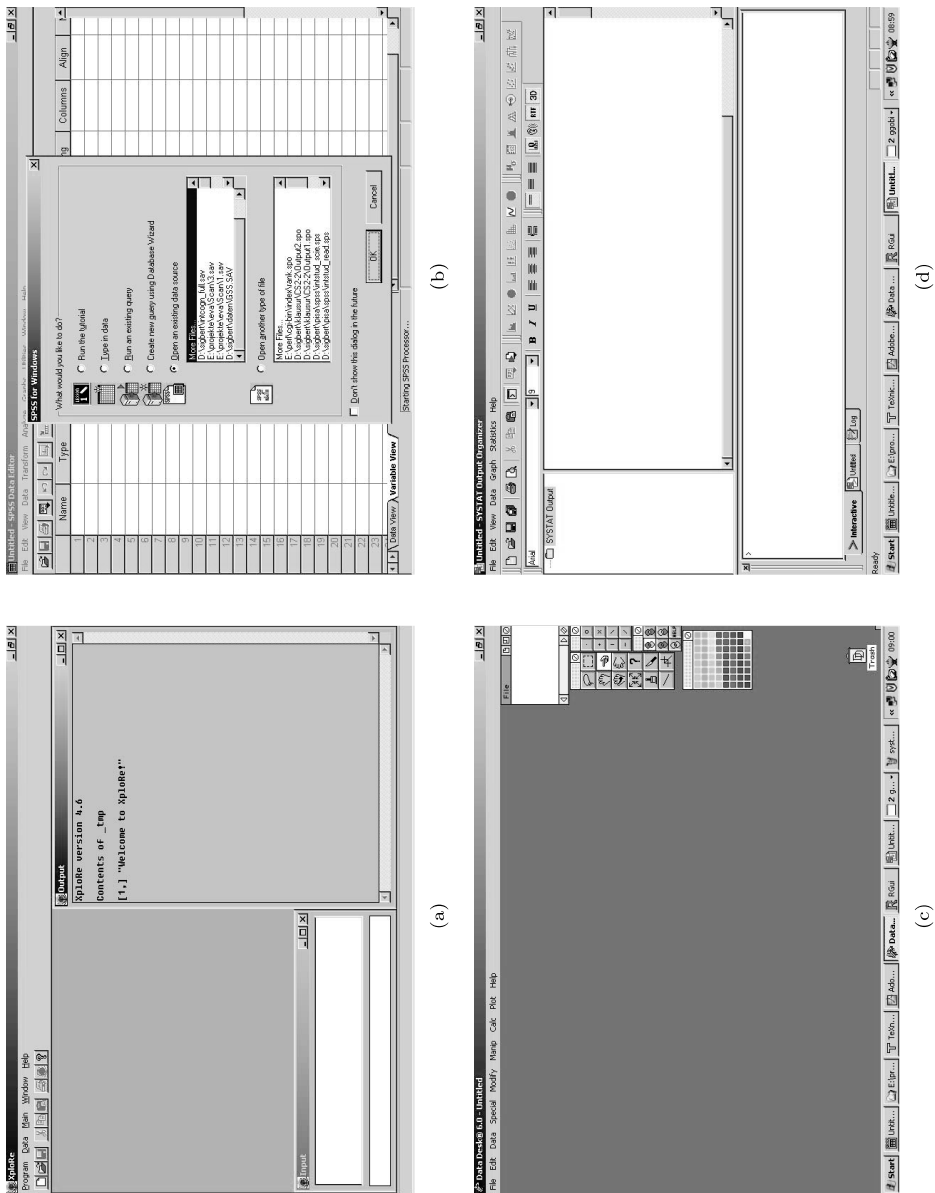


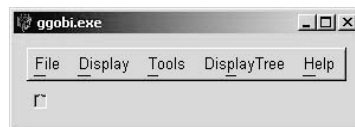
Figure 12.1. Entry screens of the statistical software packages, (a) XploRe, (b) SPSS, (c) DataDesk and (d) SYSTAT

Some packages try to help a novice user with his next task. SPSS opens, after starting the program, a dialogue box to load a dataset (see Fig. 12.1b). For R, which can load all data objects from previous sessions automatically, such feature is not necessary.

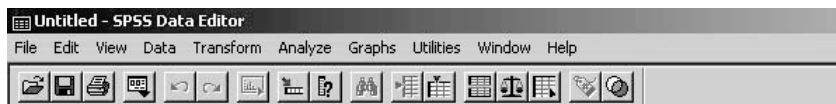
Toolbars

12.3.2

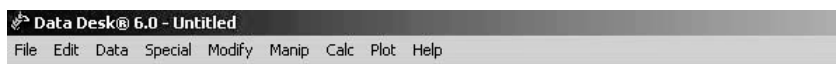
Although toolbars play a more and more important role in software nowadays, we immediately notice the sparse toolbars (see Fig. 12.2), due to the fact that we have no common set of icons. For example GGobi and DataDesk do not offer any toolbar, XploRe, SPSS and R offer only toolbars related to standard tasks, like opening, saving and printing programs, images etc. and specialized software functions. Among the considered programs only SYSTAT offers toolbars for standard statistical graphics (histogram, pie chart, boxplot, scatterplot and scatterplot matrices) and tasks (descriptive statistics, two-way-tables, two-sample t-test, ANOVA, correlations, linear regression, clustering and non-linear modeling). But to learn the meaning of the icons may take some time.



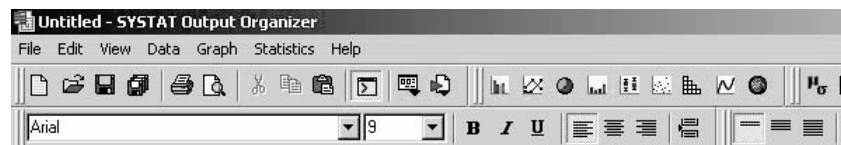
(a)



(b)



(c)



(d)

Figure 12.2. Menu bars and toolbars of the main windows of (a) GGobi, (b) SPSS, (c) DataDesk and (d) SYSTAT

Menus

12.3.3

The first parts of a software package we use are the menu bar, the menus and the menu items. Menus can give a clear structure to the methods/actions of a software package. Liebelt et al. (1982) have found that a proper organization of the menu reduces the error rate to about 50%. Most statistical software packages have a very clear separation of the tasks in the menu bar (see Fig. 12.2).

It might be distracting if the position of the menu items changes (Mitchell and Shneiderman, 1989). For unused menu items (not applicable tasks in the

current situation) it is preferable if they are grayed out and do not vanish from the menu. Statistical software packages behave very differently. The menu bar in XploRe and R changes heavily depending on the active window which can be very disturbing to the user. It would have been better to attach an appropriate menu to each window. Also in GGobi the menu changes depending on the active window: compare Fig. 12.4a to Fig. 12.2a. Nevertheless this is less disturbing to the user because additional menus appear only once in the menu bar and heavy changes take place in the items of the `Display` menu which are invisible to the user. The menu `Display` is empty after starting GGobi, but filled when a dataset is loaded.

If we create a menu hierarchy we basically have two possibilities to organize them: a small, deep hierarchy or a broad, flat hierarchy. Norman and Chin (1988) found that broad, flat hierarchies lead to a better user performance. Most software packages follow this concept intuitively, none of the software packages has a menu depth larger than four.

Several orders of menu items within menus are possible: by time, by numbering, by alphabet, by importance or by function. Card (1982) found that an alphabetical order of menu items is better than a functional order. McDonald et al. (1983) showed that the advantage of the alphabetical order is lost if each menu item consists of a one line definition rather than of one meaningful known word. Nowadays all statistical software packages prefer a functional order by separating the menu items with horizontal lines into menu item groups. But within a menu item group the order is unclear.

To achieve consistency within a menu system, the same terms should be used. If we use one word items then they should be clearly separable, like “insert” and “delete”. The exact difference between “change” and “correct” is unclear. Cyclic menus, which means we can achieve the same task by different ways through the menu hierarchy, should be avoided. Users become unsure where to find a specific action; the same problem is well known from the World Wide Web.

The approach to put the most used menu items at the top and suppress the others, may irritate the user. The irritation occurs not with the most used items, but with the items which are used less often. Their position appears to be more or less randomly. Thus, Sears and Shneiderman (1994) found that bringing only the most used items to the top of the menu is an effective technique.

For power users shortcuts, e.g. through keyboard sequences or toolbars, are very important. Often used menu items should get short shortcuts, e.g. `Ctrl+O` for open a data set, whereas rarely used shortcuts can have longer keyboard sequences. Most statistical software packages offer only the standard shortcuts coming from the Windows operating system; only GGobi offers shortcuts for different view modes. Unfortunately, we have no common sets of shortcuts for a (statistical) task. We have not even a common naming convention for menus, e.g. the statistical tasks are in the `Calc` menu in DataDesk, in the `Statistics` menu in SPSS and in the `Analyze` menu in SYSTAT.

For an effective menu system design it is helpful to log the user choices and to analyze them for improvements.

Forms and Dialog Boxes

12.3.4

The use of menus leads to another form of interaction with the user: forms and dialog boxes. Basically they should

- have a meaningful title
- use a consistent and for the user well known terminology
- group information in a consistent and meaningful way
- minimize mouse movement and jump from one item to another item in a useful way
- support error prevention
- allow for fast error correction
- provide default values, if possible
- indicate optional values clearly
- inform when the dialog or form has enough information.

Here, a very good example is SPSS. See as example in Fig. 12.3 four out of six steps for reading ASCII data into SPSS. The six dialog boxes are grouped in information about:

1. reuse of old format
2. information about the arrangement of the variables
3. information about the arrangement of the cases
4. separation between single data
5. variable formats and names
6. saving the defined format.

The forms always provide default values, show the consequence of changing a value in the bottom and allow easy navigation between forms forward and backward. We can cancel the whole operation or finish it at any time. The last form gives a clear signal when we are finished. The SPSS developers have designed all these forms and dialogs very carefully.

However, we have to keep in mind that pure statistical programming languages, like R or XploRe, will have to incorporate forms and dialog boxes in their programming language. This turns out to be a difficult task.

Windows

12.3.5

Usually statistical software packages use different windows to visualize data and output. Figure 12.4 shows a scatterplot of the variables “percentage of lower status people” and “median houseprice” of the Boston Housing data (Harrison and Rubinfeld, 1978). We easily find that the software packages have different methods to handle output. SPSS and SYSTAT have a special independent output window for text and graphic output. DataDesk, R (by default) and XploRe use the multiple document interface (MDI) coming with the Windows operating system. Actually R allows to switch between different types of handling windows (MDI/SDI). GGobi creates a complete set of independent windows.

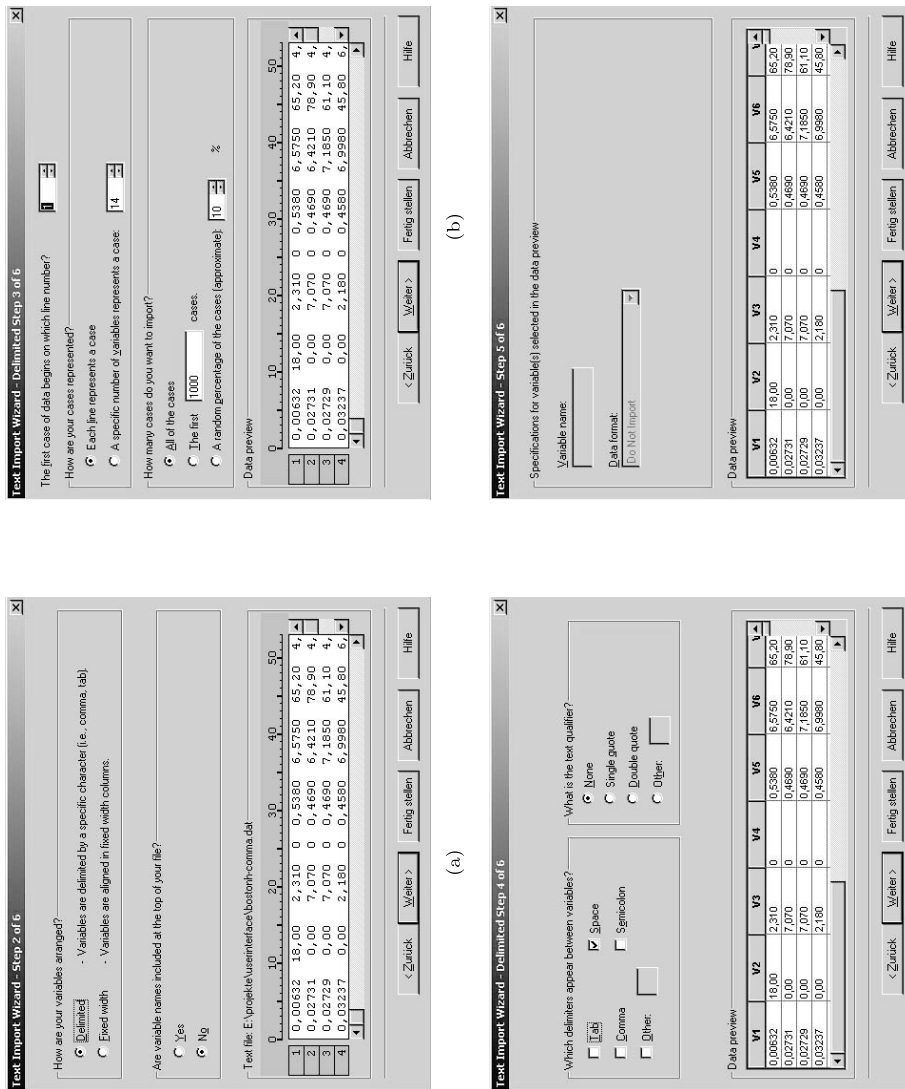


Figure 12.3. Four out of six steps of reading ASCII data in SPSS. They provide a very clear, intuitive interface even for unexperienced users for reading data

In GGobi and DataDesk the data in the windows is linked (see Fig. 12.5a). Thus interactive brushing is very easy.

A problem of some statistical software packages is that the user can easily create a lot of windows, see in Fig. 12.4 and even worse in Fig. 12.5 for DataDesk. But a large number of windows can easily irritate the user. Statistical software packages have tried to overcome this problem with different approaches: having separate graphic types, for example the scatterplotmatrix in SPSS or trellis displays in R; XploRe

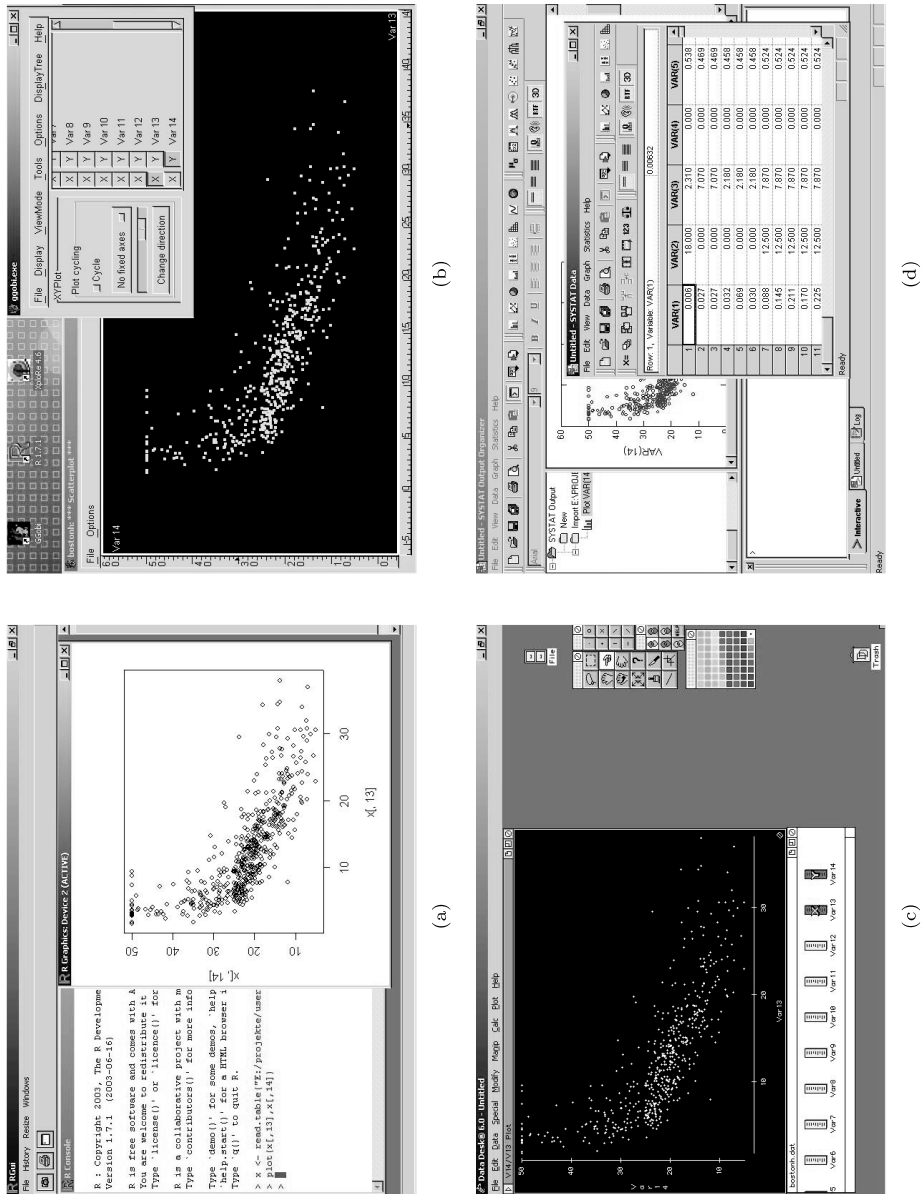


Figure 12.4. Scatterplot of the variables “percentage of lower status people” and “median houseprice” of the Boston Housing data in (a) R, (b) GGobi, (c) DataDesk and (d) SYSTAT

has a datatype for a graphical display which consists of single plots. The idea is always the same: statistical information that belongs together should be in one window. Another strategy is a virtual desktops (see the software package VanGogh in Keller, 2003) as we find them under Linux GUIs.

Power users prefer full-screen views (see Bury et al., 1985). Note that in Fig. 12.5 we tried to maximize the size of the graphics in R, SPSS and XploRe. SPSS and SYSTAT follow partially such a strategy with separating clearly between spreadsheet presentation of data and variables and output results. But Stagers (1993) has shown that users work faster with compact information on one screen rather than to scroll.

The grouping of information in a window plays an important role in GUI design. Fitts (1954) developed an effective forecasting model for the time T for a movement over a distance D to an object with width W

$$T = C_1 + C_2 \log_2(2D/W)$$

with device dependent constants C_1 and C_2 .

Maybe approaches like “The CAVE” (Cruz-Neira et al., 1993), a virtual reality environment, will lead to more screen space.

The question of the contents of the windows is related to showing windows. Tufte (1983, 1990) has shown proper and improper use of statistical graphics (see also Chap. II.11). Modern statistical techniques, like data mining, but also exploratory data analysis, has lead to principles of analysis like *get an overview, zoom, select* and *look to details*.

12.3.6 Response Times

The productivity of a user depends heavily on the response time of a software package to a user action. To achieve a good productivity we have to balance between the speed of working and the error rate. Fast response times (< 1 sec) lead to faster user work. Fast working increases the error rate because the user does not think much about the current action since he is concentrated on the responses from the software package. Slow response times (> 15 sec) lead to slower user work. Slow working decreases the error rate because the user has time to think about the next action. But if he makes a mistake he will loose time.

The right amount of the response time depends also on user experiences, e.g. if he sees that one software package reads a large dataset in 30 seconds whereas another software package needs 3 minutes for the same dataset then he will assume something has gone wrong. A power user is generally more impatient than a novice user. A partial solution to the problem of slow response times is a progress bar which shows how much time it will take till the end of the action.

Generally a user expects that simple actions, e.g. reading a small dataset, are done fast and complex actions, e.g. building a model from a large dataset, can take much longer time. The study of Martin and Corl (1986) found that the response time for a complex statistical task does not matter much for productivity, whereas the response time for a simple task (entering data) is linearly related to productivity. A variation in response times ($\pm 50\%$) does not matter much. In a set of mixed

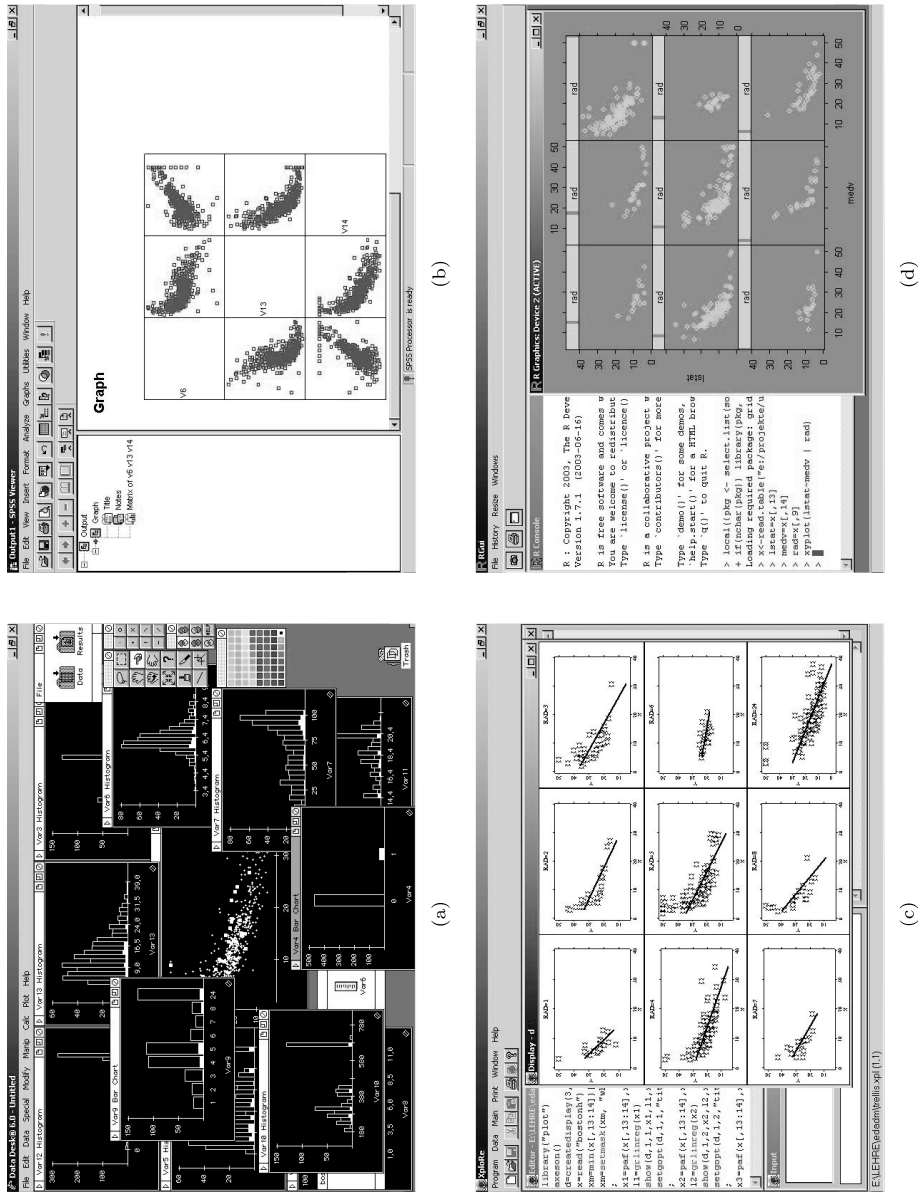


Figure 12.5. (a) Linked scatterplot, histograms and barcharts in DataDesk. (b) Scatterplotmatrix of three variables “average number of rooms”, “percentage of lower status people” and “median houseprice” in SPSS. (c) Trellis display of the variables “percentage of lower status people” and “median houseprice” conditioned on the variable “index of accessibility to radial highways” in XploRe. (d) Trellis display of the same variables in R

tasks the user balances out: he thinks about the task when the response time is slow and works fast if the response time is fast.

12.3.7 **Catching the User Attention**

In Fig. 12.4d we see that in SYSTAT the data editor stays on top, although we just created a scatterplot in the underlying output window. But the user attention is still directed to the data editor. Similar problems can be observed in other software.

Another point in GUI design we should consider is the way how we catch the attention of the user. In statistical graphics Tufte (1983, 1990) has shown how the user's attention can be redirected from the data. In the same manner a too colorful GUI may distract the user. Wickens (1992) analyzed how to catch the users attention and gave some hints:

- use 3 different fonts with 4 different sizes in 2 intensities
- use up to 4 standard colors, more colors have to be used with care
- use soft sounds when everything is okay, use hard sounds for warnings and errors.

Nowadays operating systems offer a large variety of true-type fonts, nevertheless most people use only a few fonts in their documents.

Especially the use of colors may create special problems. First, different user may combine different reactions to the same color (cultural background); second, it is known that in Europe and North America 8% of the population have problems in recognizing a color correctly. The largest problem here is the red-green blindness, both colors appear grey to such people.

The use of sound should only be an additional option. During teaching or when working in PC-Pools it will distract other users.

12.3.8 **Command Line Interfaces and Programming Languages**

In the beginning of the computer age all programs only had CLIs. One of the largest statistical software packages which has survived from these times, SPSS, still has a CLI. But it is hidden by a GUI and we can reach it via SPSS syntax editor. Statistical programming languages, like R and XploRe, are more like CLIs embedded in a GUI. Only statistical software packages like GGobi and DataDesk are real GUI software, but even DataDesk has a (visual) programming language.

In the recent past we observed that statistical packages like R or XploRe have a tendency to be split up between a GUI and a CLI. In fact on the R-Project page we find more than one GUI for R.

CLI provides some advantages compared to a pure GUI. Some manipulations, for example arithmetic transformation of data, can be done much faster with the keyboard than with the mouse.

With a programming language we can achieve a precise and compact way to manipulate and analyze data. We should be able to easily learn, read and write the programming language. Some problems that can arise are

- the design has too many objects and actions. A hierarchical approach like organizing objects and actions in libraries may help here. However, R and XploRe suffer both from an overwhelming number of packages, commands and programs.
- sometimes the names chosen for an action are too close to computer science and not to statistics. Do we *load*, *read*, *open* or *get* a dataset (see also Table 12.1)?
- inconsistent order of parameters for operations.

Modern statistical programming languages implement matrix algebra since we can easily transfer expressions, e.g. for computing the coefficients of a multiple linear regression, like $(X^T X)^{-1}(X^T Y)$ into a program (in XploRe: `inv(x' * x) * (x' * y)`). This allows for fast error correction and fast learning.

Table 12.1. Reading ASCII file with the Boston Housing data

Software	Reading ASCII data
R	<code>x <- read.table("c:/data/bostonh.dat", header=FALSE)</code>
SPSS	<code>GET DATA /TYPE = TXT /FILE = 'c:databostonh.dat' /DELCASE = LINE /DELIMITERS = " " /ARRANGEMENT = DELIMITED /FIRSTCASE = 1 /IMPORTCASE = ALL /VARIABLES = CRIM F7.2 ... MEDV F5.2 .</code>
SYSTAT	<code>IMPORT "c:/data/bostonh.dat.dat" / TYPE=ASCII</code>
XploRe	<code>x = read ("bostonh")</code>

Caroll (1982) found that hierarchical (verb-object-qualifier) and symmetric command sequences, like in Table 12.3 for linear regression, lead to the best user performance and can be easily learned and remembered. The reality in software packages is shown in the Table 12.2.

Again power users prefer rather short names whereas novice users can find actions with long names more informative. It is the best to have both available, like `DoLinearRegression` and `DoLinReg` or even `d1r`. Ehrenreich and Porcu (1982) suggest rules to make (automatic) abbreviations and Schneider (1984) proposes possible abbreviation methods:

- use a simple rule to create abbreviations
 - truncation (most preferred by users)
 - deletion of vocals (`DlnrRgrssn`)

Table 12.2. Simple linear regression with intercept between the variable “percentage of lower status people” (lstat) and the dependent variable “median houseprice” (medv) of the Boston Housing data in different statistical programming languages

Software	Linear regression commands
R	<code>res <- lm (medv ~ lstat)</code>
SPSS	<pre>REGRESSION /MISSING LISTWISE /STATISTICS COEFF OUTS R ANOVA /CRITERIA=PIN(.05) POUT(.10) /NOORIGIN /DEPENDENT lstat /METHOD=ENTER medv .</pre>
SYSTAT	<pre>REGRESS USE "c:/data/bostonh.dat" MODEL MEDV = CONSTANT + LSTAT ESTIMATE</pre>
XploRe	<code>res = linreg (lstat, medv)</code>

Table 12.3. Example of a hierarchical and symmetric command sequences in the context of linear regression

```
do linear regression
do linear regression stepwise
do linear regression forward
do linear regression backward
plot linear regression line
plot linear regression residuals
```

- use last and/or first letter
- standard abbreviation, e.g. QTY for Quantity
- phonetical abbreviation, e.g. XQT for Execute
- use a (simple) second rule for conflicts
- apply the second rule very rarely
- abbreviations with result from the second rule should have a special symbol included
- user should know both rules
- abbreviations should have a fixed length
- the software package should always use the long name, e.g. in error messages

Modern editors, e.g. the Visual Basic editor in Microsoft Office, support the writing of programs with semi-automatic command completion.

Table 12.4. Error messages in different software packages

Software	Example	Error message
XploRe	<pre>proc()=test(x) if(x=1) "true" else "false" endif endp test(0)</pre>	<p>Syntax Error in test line: 2</p> <p>Parse Error on position 5 in line 2</p>
R	if (x=1) "true" else "false"	Error: syntax error
SPSS	as in Table 12.2, just /DEPENDENT changed to /INDEPENDENT	<p>Warning</p> <p>Unrecognized text appears on the REGRESSION command. The only recognized subcommands are: Global options: DESCRIPTIVES MATRIX MISSING WIDTH; Case selection/weight: REGWGT SELECT; Variable list: VARIABLES; Equation options: CRITERIA NOORIGIN ORIGIN STATISTICS; Dependent variable(s): DEPENDENT; Equ. methods: METHOD BACKWARD ENTER FORWARD REMOVE STEPWISE TEST; Residuals: RESIDUAL CASEWISE PARTIALPLOT SAVE SCATTERPLOT OUTFILE. Text found: INDEPENDENT This command is not executed *WARNING* REGRESSION syntax scan continues. Further diagnostics from this command may be misleading – interpret with care Misplaced REGRESSION METHOD subcommand – The METHOD subcommand must follow a DEPENDENT subcommand or another METHOD subcommand. It cannot follow any other subcommand. Check for a missing DEPENDENT subcommand. Text found: METHOD</p>

A future dream is that (statistical) software understands natural language. What has proven to be valuable to the user is the generation of a report of results in natural language.

Error Messages

12.3.9

The most crucial response for a user is an error or warning message from the system. Error messages can be not very helpful, e.g. in XploRe or R `syntax error` in Table 12.4. A better solution would be to tell the user what the problem exactly was (use `x==1` instead `x=1`). But SPSS tells the user too much and the problem disappears behind the text. However, the ability of SPSS for abbreviating

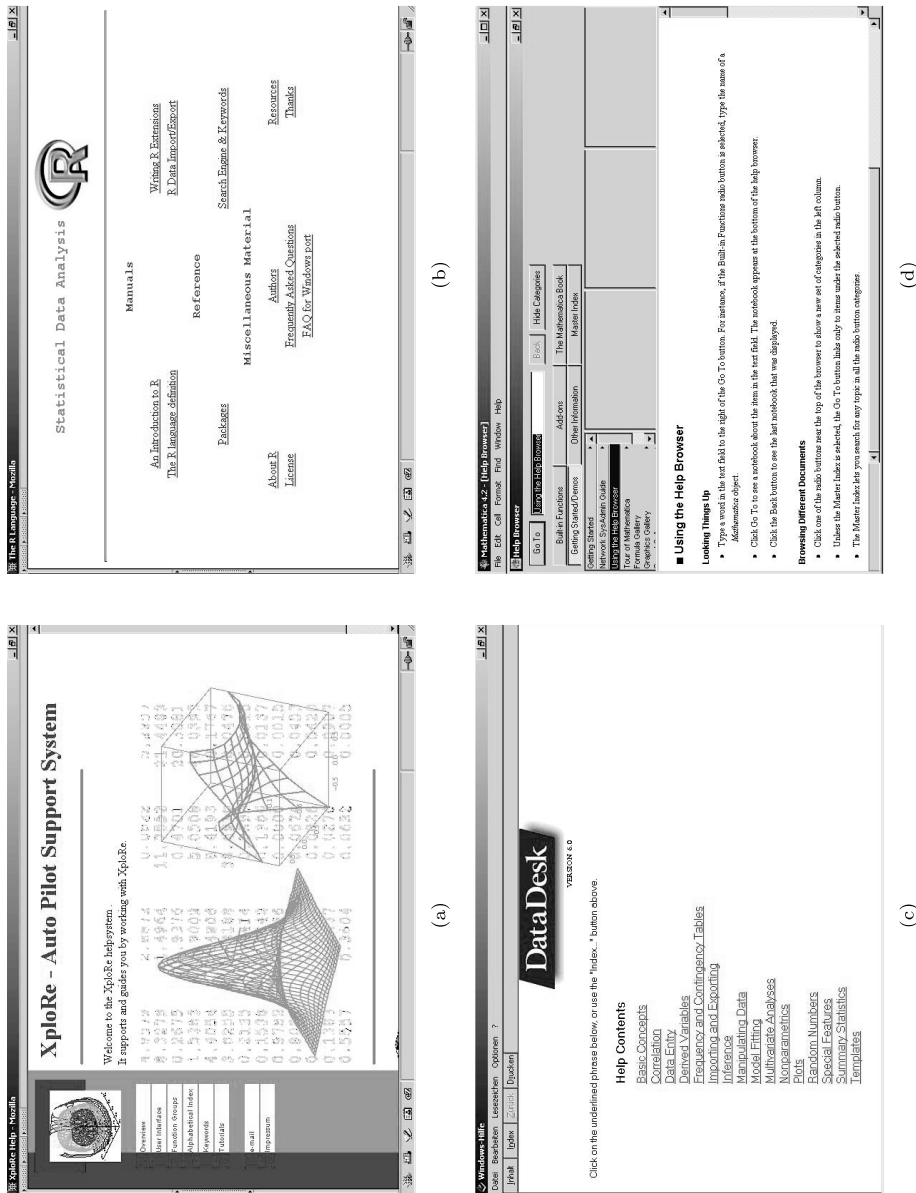


Figure 12.6. Entry screens of the help systems in (a) XploRe, (b) R, (c) DataDesk and (d) Mathematica

is impressive. From the linear regression example in Table 12.2 the parameter /NOORIGIN can be shortened to /NOO. Further shortening to /NO produces an error message.

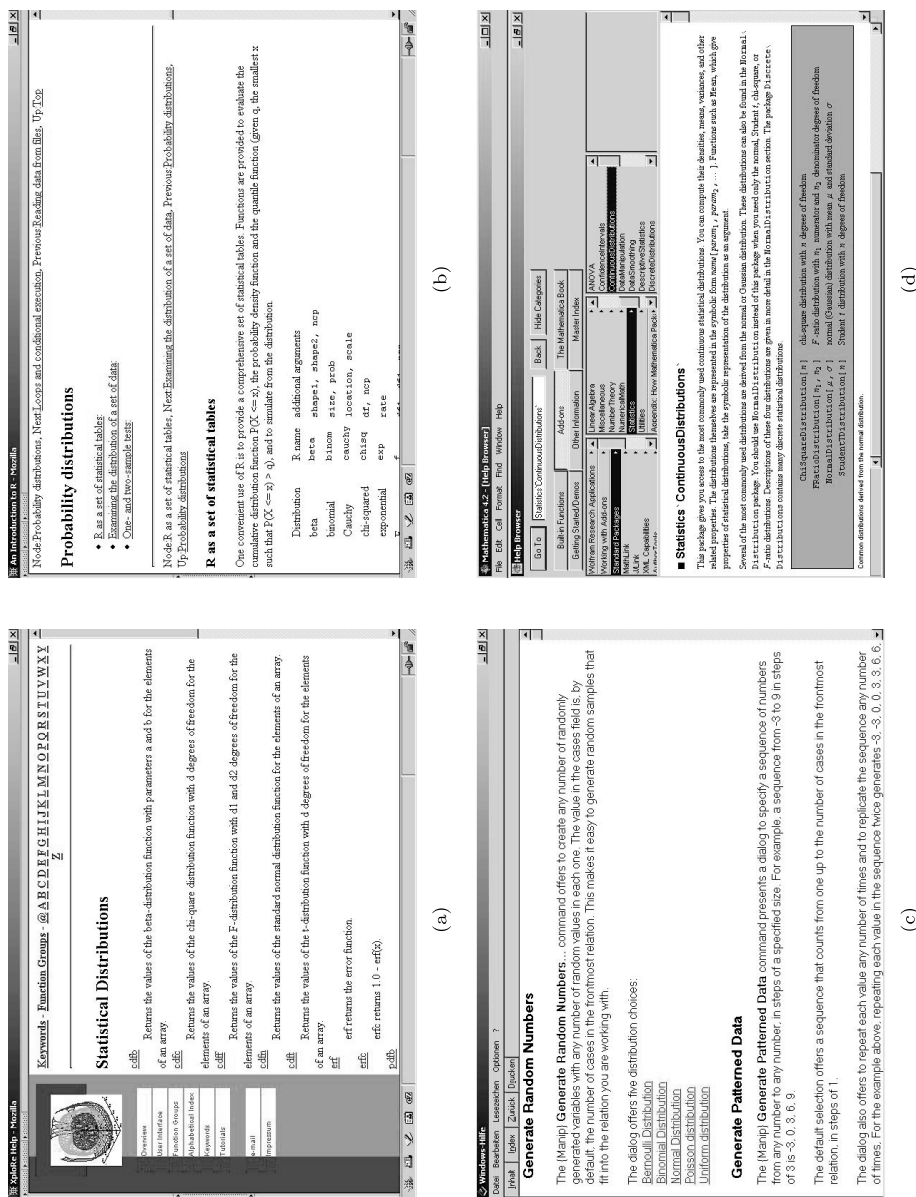


Figure 12.7. Help system entry for statistical distributions of statistical software packages, (a) XploRe, (b) R, (c) DataDesk and (d) Mathematica

The language in an error message and warning should be positive, constructive, meaningful and precise. Shneiderman (1982) found in a study that the error rate could be reduced by 28% with well constructed error messages.

Again it is a good idea to log the error message to see which ones are needed to be improved and which parts of the software package has to be improved.

12.3.10 Help System

Nowadays software is always accompanied with online help systems and tutorials, mostly HTML-based. A help system should give the user quick access to the information he needs. Depending on the type of users, they have different approaches to use a help system. Reference or alphabetical guides are useful for power users, but novice users learn most from a lot of examples. Consequently the help system of modern statistical software is mostly composed of several parts: reference/alphabetical guide, introductory tutorials, indices and a search engine.

In Fig. 12.6 we see the entry page of help systems. The variation in the software packages is large, from very sparse help systems upto detailed explanations how to use the help system.

Finding information in the help system is a crucial task. Thus good navigation, indices and search are essential for any help system. Help systems based on the windows help system, e.g. used by DataDesk, bring already the capabilities for an index and searching. Creating a good index, for a book or a help system is not easy. Especially since the developers of statistical algorithms mostly do not care much about the documentation. The quality of the help system depends heavily on the contributors to it. Maybe automated ways of analyzing tutorials and descriptions to create a hierarchy and an index can improve the help system quality.

One of useful help systems that we have seen is the help system of Mathematica which is an inherent part of it. At the top of Fig. 12.7d we see the detailed navigation, we know always where we are. Mathematica separates the information well: red background for Mathematica commands and programs and their descriptions, white background for explanations.

Generally, help systems and tutorials should have a simple language, short sentences (Roemer and Chapanis, 1982) and a consistent terminology. This has been proven more helpful to the users and most help systems follow that suggestion. It is even more important since most help systems and tutorials are written in English and the majority of the statisticians do not speak English as native language.

12.4 Outlook

There is a wishlist for the statistical user interface of a statistical software package:

- well-known icons for statistical tasks for useful toolbars
- a consistent and unified terminology for menu bars and items
- well designed dialog boxes and forms
- good editors for statistical programming languages
- a well constructed programming language

- a well designed HTML-based help system with clear structures
- a unique data format for exchanging data with other (statistical) software packages.

In the past we have observed that statistical software packages got various GUIs. Even SPSS offers a web-based interface now, like the R Web server or the XploRe Java client version. Currently we observe that statistical software packages are embedded via direct calls (Excel: XploRe with MD*ReX, R in the RDCOM server or DataDesk /XL) or via CORBA (R: Omega Hat project) in other software. In the future statistical software packages will use the GUI of the “host” software, but the problems we will encounter are the same.

Since the user interface design depends heavily on our perception and behavior, there is still a lot of experimental research necessary to find answers to the problems that occur.

Web References

DataDesk	http://www.datadesk.com
GGobi	http://www.ggobi.org
Jasp	http://jasp.ism.ac.jp
Mathematica	http://www.wolfram.com
PRIM-9 video	http://cm.bell-labs.com/cm/cms/departments/sia/video-library/prim9.html
R	http://www.r-project.org
– GUIs	http://www.sciviews.org/_rgui/
– Omega hat/RDCOM	http://www.omegahat.org
– Web	http://www.math.montana.edu/Rweb
S/S-Plus	http://www.insightful.com
SAS	http://www.sas.com
SPSS	http://www.spss.com
SYSTAT	http://www.systat.com
VanGogh	http://stats.math.uni-augsburg.de/VanGogh/
XploRe	http://www.xplore-stat.de
– Java client	http://www.xplore-stat.de/java/java.html
– MD*ReX	http://www.md-rex.com

References

- Bury, K., Davies, S., and Darnell, M. (1985). Window management: A review of issues and some results from user testing. Technical report, IBM Human factors Center Report HFC-53, San Jose, CA.

- Card, S. (1982). User perceptual mechanism in the search of computer command menus. In *Proc. Human Factors in Computer Systems*, pages 190–196, Washington D.C.
- Caroll, J. (1982). Learning, using and designing command paradigmas. *Human Learning*, 1(1):31–62.
- Cleveland, W. and Loader, C. (1996). Smoothing by local regression: Principles and methods. In Härdle, W. and Schimek, M., editors, *Statistical Theory and Computational Aspects of Smoothing*, pages 10–49. Physika Verlag, Heidelberg, Germany.
- Cruz-Neira, C., Sandin, D., and DeFanti, T. (1993). Surround-screen projection-based virtual reality: the design and implementation of the cave. In *Proc SIGGRAPH'93 Conference*, pages 135–142, New York. ACM.
- Ehrenreich, S. and Porcu, T. (1982). Abbreviations for automated systems: teaching operators and rules. In Badre, A. and Shneiderman, B., editors, *Directions in Human-Computer Interaction*, pages 111–136. Ablex, Norwood, NJ.
- Fitts, P. (1954). The information capacity of the human motor system in controlling amplitude of movement. *Journal of experimental psychology*, 47:381–391.
- Franzke, M. (1995). Turning research into practice: Characteristics of display-based interaction. In *Proc. CHI'95 Conference: Human Factors in Computing Systems*, pages 421–428. ACM.
- Hansen, W. (1971). User engineering principles for interactive systems. In *Proc. Fall Joint Computer Conference*, volume 39, pages 523–532, Montvale, NJ. AFIPS Press.
- Härdle, W. (2004). Interview with James E. Gentle. *Computational Statistics*, 19(1): 1–4. Physika Verlag, Heidelberg, Germany.
- Harrison, D. and Rubinfeld, D.L. (1978). Hedonic prices and the demand for clean air. *J. Environ. Economics and Management*, 5: 81–102.
- Keller, R. (2003). Visualizing augsburg traffic data with vangogh. Presentation at 'Workshop on Statistical Inference, Visualizing for Graphs' at Stanford University, CA., University of Augsburg, Dept. of Computer Oriented Statistics and Data Analysis.
- Liebelt, L.-S., McDonald, J., Stone, J., and Karat, J. (1982). The effect of organization on learning menu access. In *Proc. Human Factors Society, Twenty-Sixth Annual Meeting*, pages 546–550, CA.
- Marron, J. (1996). A personal view of smoothing and statistics. In Härdle, W. and Schimek, M., editors, *Statistical Theory and Computational Aspects of Smoothing*, pages 1–9. Physika Verlag, Heidelberg, Germany.
- Martin, G. and Corl, K. (1986). System response time effects on user productivity. *Behaviour and Information technology*, 5(1):3–13.
- McDonald, J., Stone, J., and Liebelt, L. (1983). Searching for items in menus: the effects of organization and type of target. In *Proc. Human Factors Society, Twenty-Seventh Annual Meeting*, pages 834–837, Santa Monica, CA.
- Miller, G. (1956). The magical number seven, plus or minus two: some limits on our capacity for processing information. *Psychological science*, 63:81–97.

- Mitchell, J. and Shneiderman, B. (1989). Dynamic vs. static menus: An experimental comparison. *ACM SIGCHI Bulletin*, 20(4):33–36.
- Norman, K. and Chin, J. (1988). The effect of tree structure on search in a hierarchical menu selection system. *Behaviour and Information Technology*, 8(2):25–134.
- Roemer, J. and Chapanis, A. (1982). Learning performance and attitudes as a function of the reading grade level of a computer-presented tutorial. In *Proc. Conference on Human Factors in Computer Systems*, pages 239–244, Washington D.C. ACM.
- Schneider, M. (1984). Ergonomic considerations in the design of text editors. In Vassiliou, Y., editor, *Human Factors and Interactive Computer Systems*, pages 141–161. Ablex, Norword, NJ.
- Sears, A. and Shneiderman, B. (1994). Split menus: effectively using selection frequency organize menus. *ACM Transaction on Computer-Human Interaction*, 1(1):27–51.
- Shneiderman, B. (1982). System message design: Guidelines and experimental results. In Badre, A. and Shneiderman, B., editors, *Directions in Human/Computer Interactions*, pages 55–78. Ablex, Norword, NJ.
- Shneiderman, B. (1998). *Designing the User Interface*. Addison Wesley Longman, Inc.
- Staggers, N. (1993). Impact of screen density on clinical nurses' computer task performance and subjective screen satisfaction. *International journal of Man-Machine Studies*, 39(5):775–792.
- Temple, Barker and Sloane, Inc. (1990). The benefits of the graphical user interface. *Multimedia Review*, pages 10–17.
- Tufte, E. (1983). *The Visual Display of Quantitative Information*. Graphics Press, Cheshire, CT.
- Tufte, E. (1990). *Envisioning Information*. Graphics Press, Cheshire, CT.
- Tukey, J., Friedman, J., and Fishkeller, M. (1973). Prim-9: An interactive multidimensional data display and analysis system. Video. ASA Statistical Graphics Video Lending Library.
- Tukey, J., Friedman, J., and Fishkeller, M. (1974). Prim-9: An interactive multidimensional data display and analysis system. Technical Report SLAC-PUB-1408, Stanford Linear Accelerator Center, Stanford, CA.
- Ulich, E., Rautenberg, M., Moll, T., Greutmann, T., and Strohm, O. (1991). Task orientation and user-oriented dialogue design. *International journal of human-computer interaction*, 3(2):117–144.
- Wickens, C. (1992). *Engineering psychology and human performance*. Harper-collins publisher.

Object Oriented Computing

II.13

Miroslav Vrius

13.1	<i>Introduction</i>	404
	First Approach to Objects	404
	Note on Unified Modelling Language	405
13.2	<i>Objects and Encapsulation</i>	405
	Benefits of Encapsulation	405
	Objects and Messages	406
	Class	406
	Object Composition	409
	Access Control	410
13.3	<i>Short Introduction to the UML</i>	410
13.4	<i>Inheritance</i>	411
	Base Class and Derived Class	412
	Generalization and Specialization	412
	Using Base Class as Common Interface	414
	Inheritance, or Composition?	417
	Multiple Inheritance	418
13.5	<i>Polymorphism</i>	418
	Early and Late Binding	419
	Implementation of the Late Binding	419
	Abstract Class	420
	Interfaces	422
	Interfaces in C++	422
13.6	<i>More about Inheritance</i>	425
	Substitution Principle	425
	Substitution Principle Revised	427

Inheritance and Encapsulation	428
13.7 <i>Structure of the Object Oriented Program</i>	431
13.8 <i>Conclusion</i>	433

In this contribution, the basic overview of the Object Oriented Programming and its usage in computation is given. Concepts of class, encapsulation, inheritance, and polymorphism are introduced. Some additional concepts like interface and Design Patterns are briefly discussed. Schematic examples in C++ are given.

Introduction

13.1

Object Oriented Programming (OOP) is a preferred methodology in contemporary software development. OOP may be considered as a continuation of the well known ideas of Structured Programming and Modular Programming. If properly used, it leads to well structured code which is easy to debug and easy to maintain.

First Approach to Objects

13.1.1

Every computer program may be considered as a software model of a real problem. It follows, that two basic domains should be taken into account during the analysis of the problem and design of the program: the *problem* domain, which is part of the real world, and the *model* domain, which is a mapping of the problem domain to the computer program.

The problem domain consists of a set of interacting objects. Selected objects of the problem domain must of course correspond to data structures in model domain and the interactions of objects in the problem domain must correspond to the operations with these data structures. That is, the interactions of objects in the problem domain will be represented by procedures and functions dealing with these data structures.

Example 1 Consider modelling the interactions of elementary particles in a detector using the Monte Carlo method. (Design and analysis of Monte Carlo experiments is discussed in depth in Chap. II.3). The problem domain of this experiment simulation consists of the detector, the particle source, the air surrounding the experimental apparatus, of many elementary particles and, of course, of a statistical file containing the simulation results. It follows, that the model of the experiment should contain a suitable representation of the detector, a suitable representation of the particle source, statistical file, etc.

1

The representation of the elementary particle source may consist of the data representing its coordinates in a given coordinate system, of a description of the spectrum of the source (i.e. of probability distributions describing the emission of various types of particles, their direction, energy and other characteristics of emitted particles) etc.

13.1.2 Note on Unified Modelling Language

To formalize object-oriented analysis and design, the Unified Modelling Language (UML) is widely used. UML consists of a set of diagrams that describe various aspects of the problem solved. We use some UML diagrams in this chapter. A short introduction to the UML is given in Sect. 13.3; full description of the UML may be found in Booch (1999).

13.2 Objects and Encapsulation

In the model domain, the term *object* denotes the data representation of the objects of the problem domain, together with the operations defined on this data.

This means that we define a data structure together with the operations with it. These operations are usually called *methods*. The object's data are denoted as *attributes*; the data and methods together are denoted as *members* of the object.

A basic rule of OOP requires that the methods should be used for all the manipulations with object's data. Methods of the object are allowed to access the data; no other access is permitted. (We shall see that under some circumstances it is acceptable to violate this rule). This principle is called *encapsulation* and is sometimes presented by the "wall of code around each piece of data" metaphor.

Note: Methods that return the value of the attribute (data member) X usually have the identifier `GetX()`; methods that set the value of the attribute X usually have the identifier `SetX()`. In some programming environments, these identifiers may be required. These methods are called *getters* and *setters*, respectively.

13.2.1 Benefits of Encapsulation

So far, there is nothing new in encapsulation: This is implementation hiding, well known from modular programming. The object may be considered as a module and the set of the methods as its interface.

The main benefit of encapsulation is that the programmer may change the implementation of the object without affecting the whole program, if he or she preserves the interface of the object. Any change of the data representation will affect only the implementation of the methods.

2 Example 2 Let's continue with the Monte Carlo simulation of the experiment with elementary particles. The object representing the detector will, of course, contain the coordinates of some important points of the detector. The first idea could be to use Cartesian coordinates; in later stage of the program development, it will be found that the spherical coordinates will suit better – e.g., because of the detector shape and program performance.

If it were allowed to manipulate the detector data directly by any part of the program, all the parts of the program that use this data should be changed. But if the encapsulation is properly applied and the data is manipulated only by the methods of the detector, all that has to be changed is the implementation of some detector methods.

Objects and Messages

13.2.2

OOP program is considered as the program consisting only of objects that collaborate by means of the *messages*. This may seem a little strange, but in this context, *to send a message to an object* means *to call a method of this object*. A message means a request for an operation on the object's data, i.e., a request to perform a method.

The object may receive only those messages for which it has corresponding methods. Sending a message that the object does not recognize causes an error. It depends on the programming language whether this error is detected in the compile time or in the run time. (In C++, it is detected in the compile time).

Class

13.2.3

Objects of problem domain may often be grouped into *classes*; one class contains objects that differ only in the values of some properties. The same holds for the objects in the model domain. The classes of objects in the problem domain are represented by user-defined data types in OOP programs called *object types* or *classes*.

The term *instance* is used to denote a variable, constant, or parameter of an object type. It is equivalent to the term *object*.

Class Members

Up to now, we have considered the class as a data type only; it serves as a template for the creation of instances. But in OOP, the class may have its own data and its own methods and may receive messages.

Data members that are part of the whole class (not of particular instances) are called *class data members* or *class attributes* and the methods that correspond to messages sent to the whole class are called *class methods*. Non-class members, attributes, as well as methods are, if necessary, denoted *instance members*.

Class data members contain data shared among all the instances of the class; class methods operate on class attributes. (From the non-OOP point of view, class data members are global variables hidden in the class, and class methods are global functions or procedures hidden in the class.)

Note: Class data members are often called *static data members* and class methods are called *static methods* in C++, Java, and some other programming languages, because they are declared using the `static` keyword in these languages.

Note: The class in C++, Java and many other OOP languages may contain definitions of other types, including other classes, as class members. Even though the so called nested classes are sometimes very useful, we will not discuss them in this article.

Note: The class in the OOP may be considered as an instance of another class; this leads to the concept of *metaclass*. Metaclass is a class that has only one instance – a class. You can find metaclasses in pure OOP languages like Smalltalk. We will not discuss the concept of metaclass here.

3 Example 3 We may suppose – at some level of abstraction – that the representation of all the particles in the Monte Carlo simulation of the experiment with the particles is essentially the same. Thus, every individual particle belongs to the *class of particles*. It follows that the model will contain the `Particle` class, and the program will contain the definition of the corresponding data type (and of course some instances of this type).

Because every particle has its own mass and velocity, the `Particle` class will contain the declaration of four data items representing particle mass and three components of the particle velocity vector. The `Particle` class should also contain methods to set and to get the values of these data items. (Later on, we will see that even other methods are necessary – e.g., a method for the interaction with the detector.)

It is also necessary to know the total number of generated particles and the actual number of existing particles in the simulation program. These numbers of the particles do not describe an individual particle and so they cannot be data members of any `Particle` instance; it is the task of the whole `Particle` class to hold these data. So they will be stored in the class attributes (because we use the C++, we may say in static attributes) of type `int`, and they will be accessed by class methods (static methods).

Definition of the `Particle` class in C++ will be as follows:

```
// Particle class definition in C++, first approach
class Particle
{
public:
    // Constructor
    Particle(double _mass, double vX,
             double vY, double vZ);
    // Instance methods
    ~Particle() { --actual; }           // Destructor
    double GetMass() { return mass; }
    void SetMass(double m){ mass = m; }
    void SetVelocity(double vX, double vY, double vZ);
    double GetVelocityX() { return velocityX; }
    // Performs the interaction with the detector
```

```

    virtual void Interact(Detector *aDetector);
    // ... and other methods
    // Class methods
    static int GetActual() { return actual; }
    static int GetTotal() {}
private:
    // Instance data members
    double mass;
    double velocityX, velocityY, velocityZ;
    // Class data members
    static int actual;
    static int total;
}; // End of the class declaration

// Definition of the static attributes
int Particle::actual = 0;
int Particle::total = 0;

// Definition of the constructor
Particle::Particle(double _mass, double vX,
                  double vY, double vZ)
: mass(_mass), velocityX(vX), velocityY(vY),
  velocityZ(vZ)
{
    ++actual; ++total;
}
// And other method definitions

```

We will not discuss the syntactic rules of the class declaration in the C++ here – this can be found in any textbook of this programming language, e.g., in Stroustrup (1998). We only note a few points.

This class contains the instance attributes `mass`, `velocityX`, `velocityY`, and `velocityZ`, and the class attributes `actual` and `total` (note the `static` keyword in their declarations). It follows that every instance of the `Particle` class will have its own data members `mass`, `velocityX`, etc. On the other hand, no instance will contain the data members `total` or `actual`. These are global variables shared by all instances and they exist even before the first instance of the `Particle` class is created and after the last one is destroyed.

The `Particle()` method is a special method called *constructor* and it serves the construction of new instances. It is invoked as a response to the message requesting the creation of a new instance of the class. (Even though it is a class method, its declaration in C++ does not contain the `static` keyword.) Its task is to initialize instance attributes. In our example, it also actualizes the values of the two class attributes.

The `~Particle()` method is another special method called *destructor* that prepares the instance for decay. In our example, it decreases the number of existing particles, because the instance for which the destructor is called will be immediately destroyed. (This is – unlike the constructor – the instance method. Note, that in garbage collected OOP languages, e.g., in Java, destructors are not used.)

13.2.4 Object Composition

One object in a program may exploit the services of another object. It may call the methods of any other independent object, or it may contain another object as a data member. The second approach is usually called *object composition*, even though it is typically implemented as composition of the classes.

Note: An object may not contain another object of the same class, of any class containing an object of the same class or of any derived class as data member. It may, of course, contain the pointers or the references to objects of any of these classes.

4 Example 4 Consider the particle source in the Monte Carlo simulation. It will be an instance of the `Source` class. For the simulation of the random processes of the emission of a particle, we will need a random number generator. The random number generator will be implemented in the program as an instance of the `Generator` class and will be based on the theory discussed in Chap. II.2. (The `Generator` class is an example of a class that has been found during the design of the `Source` class. It does not appear in the original formulation of the problem.)

This means that the `Source` class will contain an instance of the `Generator` class or a pointer to an instance of that class:

```
class Source
{
public:
    Source();
    Particle* Generate(); // Returns pointer to new
                        // particle

    // ... and other methods
private:
    Generator *gen;
    // ... and other private members
};
```

Access Control

13.2.5

Note the `private` and `public` *access specifiers* in the class declarations above. The `public` specifier declares that all the subsequent members of the class are public, i.e., they are accessible from any part of the program. The public members of the class constitute the *class interface*. The class interface usually contains only some methods and constant attributes. (Constant attributes may be accessed directly. This does not violate the encapsulation, because constant attributes cannot be changed.)

The `private` specifier means that the following members of the class are private, i.e., accessible only from the methods of the class. In other words, private members are *implementation details* of the class that can not be used by other parts of the program. Changes of private parts of the class do not change the class interface and do not affect other parts of the program.

Later on, we will see the third access specifier, `protected`. Protected members are accessible only from the methods of this class and from all the derived classes. So, they constitute the *class interface for derivation*, that may be wider than the interface of the class for common use. We will discuss the inheritance in Sect. 13.4.

The access specifiers help to implement the encapsulation. Note that in C++, as well as in many other object oriented languages, the subject of access control is the class, not the individual objects (instances). So any method called for an instance of the given class may use all private members of another instance of the same class.

Short Introduction to the UML

13.3

In Sect. 13.1.2 we have mentioned the UML. This is a modelling language based on a set of diagrams describing various aspects of the problem solved:

- The *class diagram* describes the classes used in the problem and their mutual dependencies.
- The *object diagram* describes all objects (instances) in the problem and their mutual dependencies.
- The *activity diagram* describes activities of the objects.
- The *state diagram* describes the states of objects and their possible changes and transitions.
- etc.

We will use only class diagrams in this article.

The class in the class diagram is presented as a rectangle containing the name of the class. It usually contains also the names of the methods and the names of the attributes; both may be prefixed by symbols representing their access specification (the + sign for public members, the - sign for private members and the # for protected ones). The name, the attributes and the methods are in the class icon sep-

arated by horizontal lines. If not necessary, attributes and methods may be omitted. Figure 13.1 shows the icon of the `Source` class as we have designed it in Sect. 13.2.4.

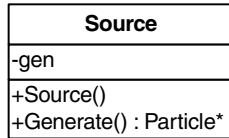


Figure 13.1. The full UML icon of the `Source` class

Associations (i.e., any relations) among classes in UML class diagrams are represented by lines connecting the class icons ended by arrows; as a description, the multiplicity of the relation may be given. For example, the number appended to the line connecting the `Source` and the `Particle` classes in Fig. 13.2 express the fact that one particle source may emit any number of particles. The number appended to the line connecting the `Source` and the `Generator` class express the fact that one source uses only one random number generator.

Object composition is expressed by the arrow ending with a filled diamond. Fig. 13.2 shows relations among the `Source`, `Particle`, and `Generator` classes. Simplified class icons are used.

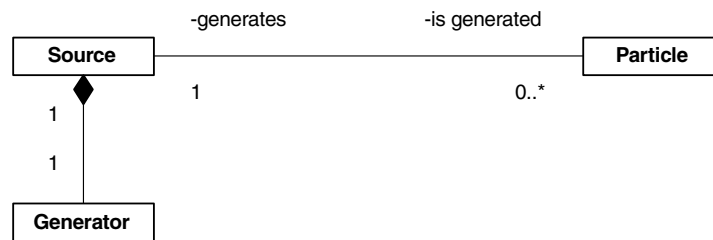


Figure 13.2. Relations among the `Source`, `Particle` and `Generator` classes

13.4

Inheritance

Inheritance is a very powerful tool used in OOP to derive new classes from existing ones. First, look at an example.

5 Example 5 Investigating our Monte Carlo simulation more deeply, we find, that various types of elementary particles can be involved: photons, neutrons, neutrinos, etc.

On the one hand, we may conclude that one common data type, the `Particle` class, is sufficient for the representation of all the different particles, because they have many common features:

- Every particle has a velocity vector,
- every particle has a mass, a spin, and electrical charge,

-
- every particle has its halftime of decay,
 - every particle may interact with the detector,
 - etc.

On the other hand, the way of the interaction with the detector is substantially different for different types of the particles. In some cases, it is described by mathematical formulae, in other cases it is described by measured data only. It follows that the operation representing the interaction of the particle in the detector must be implemented in a different way for different types of the particles, and this leads to the conclusion that different types of simulated particles have to be represented by different object types in the program; but these types share many common properties.

This situation – closely related, but different classes – can be expressed in the program model: OOP offers the mechanism of *inheritance*, which is the way of deriving one class from some other one (or other ones).

The class a new type is derived from is usually called the *base class*.

Base Class and Derived Class

13.4.1

The derived class *inherits* all the public and protected members of its base class or classes. This means that the derived class contains these members and may access them without any constraints. Private members are not inherited. They are not directly accessible in the derived class; they may be accessed only by the access methods inherited from the base class.

The derived class may add its own data members and methods to the inherited ones. The derived class may also redefine (*override*) some of the methods defined in the base class. (To override a method in a derived class means to implement a different response to the same message in the derived class.) In this case, the signature, i.e., the identifier, the return type, the number, and the types of the parameters of the overriding method in the derived class should be the same as the signature of the overridden method in the base class.

No members of the base class may be deleted in the inheritance process.

The set of all the classes connected by inheritance is usually called the *class hierarchy*.

Note that in some programming languages there are exceptions to the above rule. For example, the constructors, destructors, and overloaded assignment operators are not inherited in C++. Instead, the constructor of the derived class always calls the base class constructor and the destructor of the derived class always calls the base class destructor. The same holds for the default assignment operator of the derived class. Of course, this may be considered as a generalized form of inheritance.

Generalization and Specialization

13.4.2

The base class always represents a concept that is more general and more abstract, than the concept represented by the derived class; it follows that the de-

derived class represents a more specialized concept than the base class. In other words, the derived class always represents a *subclass* – or a subtype – of its base class. *Any instance of the derived class is also considered to be an instance of the base class.*

The interface of the base class is a subset of the interface of the derived class.

Consequently, an instance of the derived class may be used everywhere where an instance of the base class is expected. This rule may significantly simplify the operation with instances of many similar classes.

In the UML class diagram, the inheritance is represented by an arrow ending with triangle (not filled). The arrow leads from the derived class to the base class.

6 Example 6 Consider the `Particle` class in our Monte Carlo simulation. This is a general concept that can be used to describe common features of all the particles involved. But in the simulation, concrete types of particles – e.g., protons, electrons, etc. – will be used.

Consequently, we will use the `Particle` class as the base class of the particles hierarchy that will contain the common data members and the common methods of all the particles. All the classes representing concrete particle types will be derived from the `Particle` class – see Fig. 13.3.

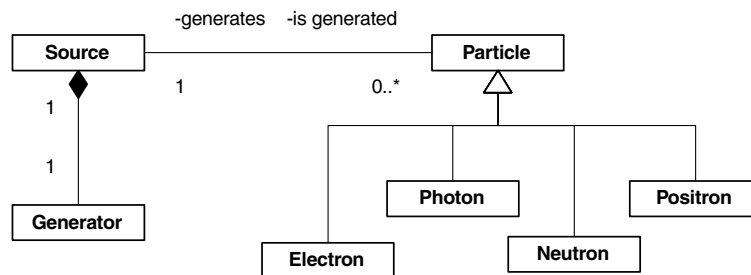


Figure 13.3. The `Particle` class as a base class of concrete particle types

We will show here only the declaration of the `Electron` class.

```

class Electron: public Particle
{
public:
    Electron();
    void SetCharge(double _charge) { charge = _charge; }
    virtual void Interact(Detector *aDetector);
private:
    double charge;
};
  
```


The declaration of the `Electron` class contains only the declaration of the constructor, two access methods and one data member. Nevertheless, the methods `GetvelocityX()`, `SetVelocityX()`, `GetActual()` etc., inherited from the base class, may be called for any instance of this class. This class changes – overrides – the implementation of the `Interact()` method.

On the other hand, data members `mass`, `velocityX`, `actual`, etc. are not directly accessible in the `Electron` class. These data members are in the base class declared as private and the derived class must manipulate them only using the public access methods. So the following fragment of the definition of the `Electron::Interact()` method is incorrect:

```
// Error: velocityX, velocityY and velocityZ
// are inaccessible in the Electron class.
void Electron::Interact(Detector *aDetector)
{
    double velocity = sqrt(velocityX*velocityX +
                          velocityY*velocityY + velocityZ*velocityZ);
    // ... and so on ...
}
```

The correct form uses the access methods inherited from the base class:

```
// Correct form of the previous code fragment
void Electron::Interact(Detector *aDetector)
{
    double velocity = sqrt(GetVelocityX()*GetVelocityX()
                          + GetVelocityY()*GetVelocityY()
                          + GetVelocityZ()*GetVelocityZ());
    // ... and so on ...
}
```

Of course, this is sometimes inconvenient. If we change the access specifiers of these data members in the base class to `protected`,

```
// Particle class definition revised
class Particle
{
public:
    // Public members as before
protected:
    // Instance data members
    double mass;
    double velocityX, velocityY, velocityZ;
    // Class data members
    static int actual;
    static int total;
};
```

the problems with access to data members will not appear. On the other hand, this violates the encapsulation of the base class and it may have a negative impact on the clarity and maintainability of the program.

13.4.3 Using Base Class as Common Interface

As stated above, instances of derived classes may be used anywhere instances of the base class are expected. This gives us very powerful tool to deal with the objects of the classes derived from the same base class in a uniform manner.

7 Example 7 In the Monte Carlo simulation of the particle experiment, we may first store all the emitted particles in a suitable container, exclude particles, that do not hit the detector etc., and after that preprocessing, let the remaining particles interact with the detector. Consider the following fragment of code:

```
const int N = 1000000;
    // Number of particles to emit
vector<Particle*> store;
    // Store for particles
Source scel;
    // Particle source
Detector det;
    // Detector in the experiment
for(int i = 0; i < N; i++)
    store.push_back(scel.Generate());
// ... some preprocessing of the set of the particles
for(int i = 0; i < store.size(); i++)
    store[i] -> Interact(det);
```

The `store` variable is a vector (dynamically sized array) of pointers to `Particle`, the base class of all the elementary particles involved. This allows us to store pointers to instances of any class derived from the `Particle` class in this container.

The expression

```
store[i] -> Interact(det);
```

represents the call of the `Interact()` method of the particle pointed to by `store[i]`. (In fact, this statement calls the method of the `Particle` class. To ensure that the method of the actual class of the particle is called, the method needs to be declared with the `virtual` specifier. This will be discussed later in Sect. 13.5, *Polymorphism*.)

This example demonstrates that the base class defines the common interface for all the derived classes.

Technical Note

The conversion from the derived class to the base class is automatic, but in some situations deserve special attention. Consider the following assignment:

```
Electron e;           // Non-dynamical instances
Particle p;
p = e;
```

After this statement has been executed, the `p` variable will contain an instance of the `Particle` class, not an instance of the `Electron` class! The reason is simple: The declaration

```
Particle p;
```

reserves store only for the `Particle` instance, so there is no place for the additional data members declared in the `Electron` class. The only way how to execute the assignment is to convert the derived class instance `e` to the base class first.

The same problem arises in the case of passing function arguments by value. Having the function

```
void process(Particle p);    // Pass by value
```

it is possible to write

```
process(e);                 // e is Electron
```

but the instance `e` of type `Electron` will be first cast (converted) to the `Particle` type. This cast leads to the loss of information.

These problems may be avoided if we use dynamical instances only. If we rewrite the preceding declarations into the form

```
Electron *ep = new Electron; // Dynamical instances
Particle *pp;
pp = ep;                    // OK
```

the `pp` variable will still contain the pointer to the instance of the `Electron` class. (The type of the pointer is converted, not the type of the instance.)

The parameter of the `process()` function should be passed by the pointer or by the reference. In both cases, the original instance is accessible in the function body and no information is lost.

In Java, C# and other OOP languages, that use dynamical instances of the object types only and manipulate them by references, these problems do not appear.

13.4.4 Inheritance, or Composition?

In some cases, it is not clear, whether a new class should be derived from some suitable class by inheritance or whether object composition should be used.

There are two questions that should be answered in this case:

- *Is the new class a special case of the base class proposed?*
- *Has the new class a data member of the proposed class?*

This is known as the *IS A – HAS A* test. Only if the answer to the first question is *yes*, the inheritance may be considered, otherwise the composition should be used.

8 Example 8 Consider the `Source` class, representing the source of elementary particles in the Monte Carlo simulation. It will be based on some generator of random numbers represented in our program by the `Generator` class. In other words, the `Source` seems to be the `Generator` class with some added functionality. Should we derive the `Source` class from the `Generator` class?

If we apply the *IS A – HAS A* test, we find that the particle source is not a special case – a subclass – of the random number generator. It uses the random number generator, so it may contain it as a data member, however, the inheritance should be avoided in this case.

Consider for an instant that we use the inheritance to derive the `Source` class from the `Generator` class,

```
// Wrong use of the inheritance
class Source: public Generator
{
    // Body of the class
}
```

This would mean that we can use a `Source` instance everywhere the `Generator` instance is expected. But in the Monte Carlo simulation, the random number generator is necessary even in other parts of the program, e.g. for the determination of the interaction type, for the determination of the features of the secondary particles resulting from the interaction (if any) etc. However, in these parts of the program, the `Source` class may not serve as the random number generator.

Such a design of the `Source` class may cause that some typing errors in the program will not be properly detected and some mysterious error messages during the compilation will be reported; or even worse – it may lead to runtime errors hard to discover.

Multiple Inheritance

13.4.5

The class may have more than one base class; this is called *multiple inheritance*. The class derived from multiple base classes is considered to be the subclass if all its base classes.

Multiple inheritance may be used as a means of the class composition.

This feature is supported only in a few programming languages – e.g., in C++ (see Stroustrup, 1998) or in Eiffel (see Meyer, 1997). In Java, C#, Object Pascal and some other languages it is not supported. Multiple inheritance poses special problems, that will not be discussed here.

Polymorphism

13.5

At the end of the previous section, we have seen that instances of many different classes were dealt with in the same way. We did not know the exact type of the instances stored in the `store` container; it was sufficient that they were instances of any class derived from the `Particle` class.

This feature of the OOP is called *polymorphism*. Polymorphism means that instances of various classes may be used in the same way – they accept the same messages, so their methods may be called without any regard to the exact type of the instance. In some programming languages (e.g., in Java), this is automatic behavior of the objects (or of their methods), in some programming languages (e.g. in C++) this behavior must be explicitly declared.

There are at least two ways to achieve polymorphic behavior: The use of the inheritance and the use of the interfaces. The interfaces will be discussed in Sect. 13.5.4.

Example 9 Let's consider once again the example given at the end of the *Inheritance* section. The expression `store[i]` is of type "pointer to the `Particle` class", even though it in fact points to an instance of the `Electron`, `Photon`, or some other derived class.

9

It follows that the statement

```
store[i] -> Interact(det); // Which method is called?
```

might be interpreted as the call of the `Particle::Interact()` method, even though it should be interpreted as the call of the `Interact()` method of some derived class.

13.5.1 Early and Late Binding

The previous example shows that there are two possible approaches to the resolution of the type of the instance for which the method is called, if the pointer (or reference) to the instance is used:

- *Early binding*. The type of the instance is determined in the compile time. It follows that the static (declared) type of the pointer or reference is used. This is the default for all methods in C++, C#, or Object Pascal.
- *Late binding*. The type of the instance is determined in the run time. It follows that the actual type of the instance is used and the method of this type is called. This is always used for the methods in Java. In C++, the `virtual` keyword denotes the methods using the late binding.

Late binding gives the class polymorphic behavior. On the other hand, late binding is less effective than early binding, even though the difference may be negligible. (In C++ on PCs, the difference between the late and the early binding is usually one machine instruction per method call.)

Any method that might be overridden in any of the derived classes should use the late binding.

Note: In C++ and other OOP languages in which the late binding must be declared, the classes containing at least one virtual method are called *polymorphic classes*. Classes without any virtual method are called *non-polymorphic classes*. In languages like Java, where all the methods use late binding by default, all the classes are polymorphic.

13.5.2 Implementation of the Late Binding

In this subsection, some low level concepts will be discussed. They are not necessary for understanding the basic concepts of the OOP, but they can give better insight in it.

We will explore one the common way of implementation of the late binding, i.e., of the determination of the actual type of the instance for which the method is called.

This is based on the so called *virtual method tables*. The virtual method table (VMT) is the hidden class data member that is part of any polymorphic class. Any polymorphic class contains exactly one VMT. (The hidden class member is a class member the programmer does not declare – the compiler adds it automatically.)

The VMT is an array containing pointers to all the virtual methods of the class. Any derived class has its own VMT that contains pointers to all the virtual methods (even those that are not overridden in this class). The pointers to the virtual methods in the VMT of the derived class are in the same order as the pointers to corresponding methods in the base class.

Any instance of the polymorphic class contains another hidden data member – the pointer to the VMT. This data member is stored in all the instances at the same place – e.g. at the beginning.

The method call is performed in the following way:

1. The program takes the instance, for which the method is called.
2. In the instance, it finds the pointer to the VMT.
3. In the VMT, it finds the pointer to the method called. In all the VMTs this pointer is in the same entry; e.g., the pointer to the `Interact()` method might be in the VMT of the `Particle` class and in the VMTs of all the classes derived from the `Particle` in the first entry.
4. The program uses this pointer to call the method of the actual class of the instance.

Figure 13.4 illustrates this process for the `Particle` base class and two derived classes. The values stored in the VMT are set usually at the start of the program or when the class is loaded to the memory. The values of the pointer to the VMT in the instances are set automatically by the constructor.

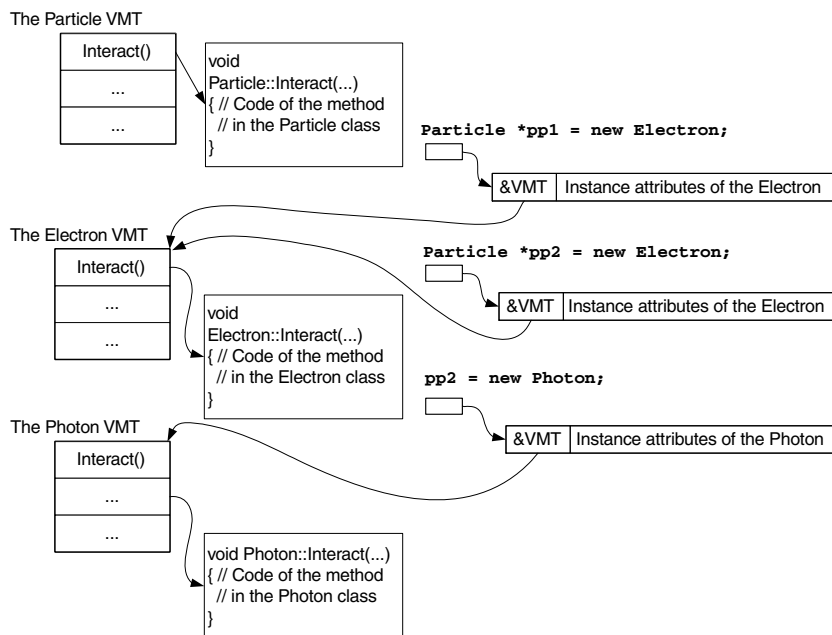


Figure 13.4. Typical implementation of the late binding

Abstract Class

13.5.3

In some cases, the base class represents such an abstract concept that some operations with instances of this class cannot be implemented. Nevertheless, at least the

stub of the corresponding method should be present in the class, because this class is used as a base class and determines the common interface for all the derived classes.

In OOP such a class is called the *abstract class* and such an operation is called the *abstract method*. Abstract methods have no implementation (no method body).

It is not allowed to create instances of the abstract classes and it is not allowed to call the abstract methods. It is of course possible to define pointers or references to abstract classes.

The abstract classes serve as base classes. If the derived class does not implement any of the inherited abstract methods, it will be abstract like the base class. The abstract class

- defines the interface of the derived classes,
- provides the implementation of non-polymorphic methods for the derived classes, and
- offers a default implementation of non-abstract polymorphic (virtual) methods.

Note that the abstract classes are italicized in the UML class diagrams – see e.g. the *Particle* class in Fig. 13.5.

10

Example 10 Consider the `Particle` class defined above. How could the `Interact()` method be implemented?

For the derived classes, the situation is clear: If it is, e.g., the `Photon`, the interaction could be the photoelectric effect, the Compton scattering, or some other interaction known to particle physicists; probabilities of these phenomena are determined according to their total effective cross sections. For the other derived classes, there are other well defined possibilities that can be expressed in the program code.

However, there is no general interaction, that could be used to implement the `Interact()` method of the general `Particle`. On the other hand, this method must be declared in the `Particle` class as the part of the common interface of derived classes. If we omit it, the statement

```
store[i] -> Interact(det);
```

will not compile, because `store[i]` is the pointer to the `Particle` class that does not contain such a method.

So, the `Interact()` method should be declared as abstract (in C++, the abstract methods are called *pure virtual methods*). In the following revision of the `Particle` class, we omit all other parts of that class that are unchanged.

```
// Particle as an abstract class
class Particle
{
```

```

public:
    // Pure virtual method
    virtual void Interact(Detector *aDetector) = 0;
    // All other public members as before
protected:
    // All data members as before
};

```

Interfaces

13.5.4

The *interface* may be defined as a named set of methods and constants. This set may be empty.

The interface represents a way to achieve the polymorphic behavior; it is an alternative to inheritance. This concept is relatively new in OOP; it was first widely used in the Java language.

In languages that support interfaces, any class may declare that it *implements* the given interface. This means that the class will supply the implementations (bodies) of the methods in the interface.

Note the terminological difference: Even though interfaces are syntactically similar to the classes that contain only public abstract methods, they are not *inherited*, but they are *implemented*. In programming languages, that support interfaces, any class may implement many interfaces, even if the language does not support multiple inheritance.

The interface represents the type. If class C implements interfaces I1 and I2, any instance of this class is an instance of type C and also an instance of type I1 and of type I2.

The interface is usually represented by a small circle connected to the class icon in the UML class diagrams (see Fig. 13.5). It may also be represented by a class-like icon marked by the `<<interface>>` label (“stereotype”). Implementation of the interface is represented by a dashed arrow pointing to the implementing class (see Fig. 13.7).

Interfaces in C++

13.5.5

As we have chosen C++ as the language of examples, it is necessary to cover briefly the interfaces in this language. C++ does not support interfaces directly; nevertheless, interfaces may be fully simulated by abstract classes that contain only public abstract (pure virtual) methods, and the interface implementation may be substituted by the inheritance. This will be demonstrated by the following example.

Example 11 The Monte Carlo simulation may be time-consuming and it would be convenient to have the possibility to store the status of the simulation into a file, so that the computation might be interrupted and continued later.

11

It is clear that all the generated but not yet processed particles should be stored. The status of the particle source, and consequently the status of the random numbers generator, should be stored, too. This is necessary especially for debugging, because it ensures that we could get the same sequence of random number in repeated runs of the program, even if they are interrupted.

It follows that we have at least two different object types belonging to different class hierarchies that have a common feature – they will be stored in a file and later will be restored into their original state. It follows that all the classes involved should have suitable methods, e.g., `store()` and `restore()`.

The simulated experiment is represented by the `Experiment` class in the program and to store the experiment status is the task of this class; so we would like to implement in this class the `storeObject()` method to store objects passed as arguments. It follows that all the parameters – all the objects stored – should be of the same type.

The solution of this dilemma – the method requires objects of the same type as parameters, but we have objects of at least two distinct types belonging to different class hierarchies – is to use a suitable interface that contains the `store()` and `restore()` methods. We will use the `Storable` identifier for this interface. The `Source`, `Generator` and `Particle` classes should be modified as follows:

```
// Interface simulation in C++
class Storable
{
public:
    virtual void store(ostream&) = 0;
    virtual void restore(istream&) = 0;
};

class Generator: public Storable
    // Interface implementation
{
public:
    virtual void store(ostream& out)
        { /* Store the generator */ }
    virtual void restore(istream& in)
        { /* Read the generator and reconstruct it */ }
    // ... Other methods and attributes as before
};

class Source: public Storable
    // Interface implementation
{
public:
    virtual void store(ostream& out)
        { /* Store the source */ }
    virtual void restore(istream& in)
```

```

    { /* Read the source from the file
      and reconstruct it */ }
    // ... Other methods and attributes as before
};

class Particle: public Storable
    // Interface implementation
{
public:
    virtual void store(ostream& out)
    { /* Store the particle */ }
    virtual void restore(istream& in)
    { /* Read the particle from the file
      and reconstruct it */ }
    // ... Other methods and attributes as before
};

```

(`ostream` and `istream` are base classes for output and input data streams in the standard C++ library). Figure 13.5 shows the revised UML class diagram of these classes.

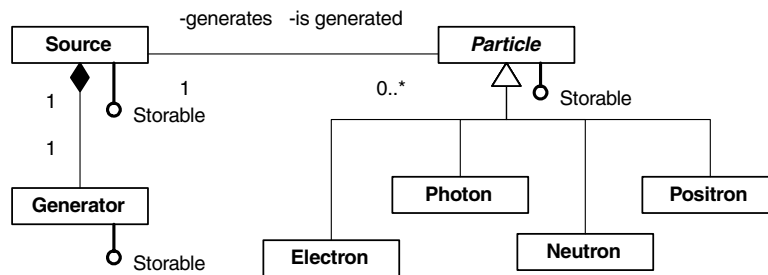


Figure 13.5. The classes implementing the `Storable` interface

Note that the `Particle` class is abstract, so it need not override the methods of the `Storable` interface. The classes representing the concrete particle types, `Electron` etc., inherit the implementation of the `Storable` interface; thus it is not necessary to declare this fact. Of course, if a derived class is not abstract, it must override the methods of this interface.

Implementation of the `Storable` interface allows us to define the method `Experiment::storeObject()` as follows:

```

void Experiment::storeObject(Storable& obj,
                             ostream& out) {
    obj.store(out)
}

```

`Storable` interface serves as the common type of all the storable objects – particles as well as random number generators – in the program. This gives us the possibility to treat all these objects in our program in a uniform way.

13.6 More about Inheritance

Inheritance can be easily misused and this is often counterproductive. Poorly designed inheritance hierarchies lead to programs that are difficult to understand, contain hard-to-find errors, and are difficult to maintain. In this section, we give some typical examples.

13.6.1 Substitution Principle

In Sect. 13.4, *Inheritance*, we have seen that any instance of any derived class may be used where an instance of the base class is expected. This is sometimes called *the substitution principle*.

As far as we have seen, this is a syntactic rule: If you follow it, the program compiles. But we already know that for reasonable use of the inheritance, the derived class must be a specialization of the base class. Let's investigate more deeply, what it means.

“Technical” Inheritance

This problem is similar to the problem we have seen in Sect. 13.4.4. We have two related classes, say A and B, and class B contains all the members of A and some additional ones. Is it reasonable to use A as a base class of B?

Of course, this situation indicates, that B *might be* really derived from A. But this is indication only that cannot replace the IS A – HAS A test. In Sect. 13.4.4, we have seen an example that leads to object composition. Here we give another example that will be solved by inheritance.

12 Example 12 Consider the particles in our Monte Carlo simulation. The interaction of electrically charged particles in the detector substantially differs from the interaction of uncharged particles, so it would be convenient to split the class of all the particles into two subclasses, one for the uncharged particles and the other for the charged ones.

The class representing the charged particles contains the same data members as the class representing the uncharged particles plus the `charge` attribute and the methods `setCharge()` and `getCharge()` to manipulate the charge. This might lead to the idea to define the `Uncharged` class representing the uncharged particles and use it as a base for the `Charged` class representing the charged particles. These two classes will serve as base classes for the classes representing concrete particle types (Fig. 13.6a).

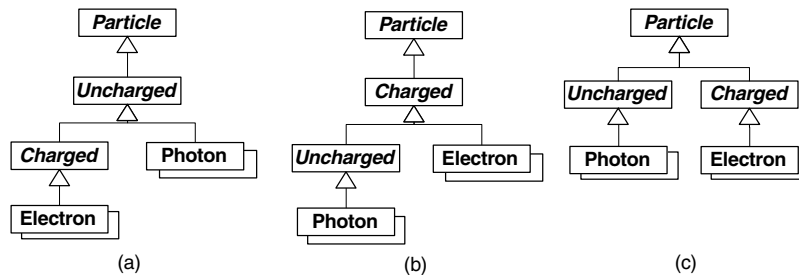


Figure 13.6. Class hierarchies discussed in Sects. 13.6.1 and 13.6.2. Only (c) is correct

This class hierarchy design is incorrect and leads to problems in the program. Suppose the following two declarations:

```
list<Uncharged*> ListOfUncharged;
Electron e; // Electron is charged particle
```

The `ListOfUncharged` variable is a double-linked list of pointers to the `Uncharged` instances. If the `Charged` class were derived from the `Uncharged` class, it would be possible to insert any charged particle into this container of uncharged ones. The following statement would compile and run (of course, the results would be unpredictable):

```
ListOfUncharged.push_back(&e); // It compiles...
```

The reason of this problem is evident – *the charged particle is not a special case of the uncharged particle* (the IS A test), so this inheritance is not applicable.

“Logical” Inheritance

Here we will show that the IS A – HAS A test may be insufficient in some cases. First, consider the following example.

Example 13 We will continue with the analysis of the Monte Carlo simulation of the charged and uncharged particles. The uncharged particles may be considered as a special case of the charged particles with the electric charge set to zero. Consequently, it seems to be logical to derive the `Uncharged` class from the `Charged` class (Fig.13.6b).

However, no member of the base class may be excluded from the derived class in the inheritance process. So the derived class, `Uncharged`, will contain the charge attribute and both the access methods. In order to ensure that the charge is zero, we have to override the `setCharge()` method so that it always sets the charge value to zero,

```
void Uncharged::setCharge(double ch) {  
    charge = 0.0;           // Parameter value not used  
}
```

Nevertheless, this construction may fail. Consider the following function:

```
void process(Charged& cp){  
    const double chargeValue = 1e-23;  
    cp.setCharge(chargeValue);  
    assert(cp.getCharge() == chargeValue);  
    // And some other code...  
}
```

This is correct behavior of the `process()` function: It expects a charged particle, changes its charge to some predefined value and tests whether or not this change succeeded. If the argument is really a charged particle, it works.

However, the classes representing the uncharged particles, e.g., `Photon`, are derived from the `Uncharged` class and this class is derived from the `Charged` class, so the following code fragment compiles, but fails during the execution:

```
Photon p;           // Uncharged particle  
process(p);        // Assertion fails...
```

This example shows, that even if the IS A test succeeds, it does not mean that the inheritance is the right choice. In this case, the overridden `setCharge()` method violates the contract of the base class method – it does not change the charge value.

13.6.2 **Substitution Principle Revised**

The preceding example demonstrates that under some circumstances the `Uncharged` class has significantly different behavior than the base class, and this leads to problems – even to run time errors.

This is the rule: Given the pointer or reference to the base class, if it is possible to distinguish, whether it points to an instance of the base class or of the derived class, the base class cannot be substituted by the derived class.

The conclusion is, that the substitution principle is more than a syntactic rule. This is a constraint imposed on derived classes, that requires, that the derived class instances must be programmatically indistinguishable from the base class instances, otherwise the derived class does not represent a subtype of the base class.

This conclusion has been originally formulated by Liskov (Liskov, 1988; Martin, 1996) as follows:

What is wanted here is something like the following substitution property: If for each object o_1 of type S there is an object o_2 of type T such that for all programs P defined in terms of T, the behavior of P is unchanged when o_1 is substituted for o_2 , then S is subtype of T.

Example 14 Let's finish the charged and uncharged particles problem. We have seen that the `Charged` and `Uncharged` classes may not be derived one from the other. To avoid both kinds of problems, it is necessary to split the hierarchy and to derive both classes directly from the `Particle` class:

14

```
// Proper Particle hierarchy
class Charged: public Particle { /* ... */ };
class Uncharged: public Particle { /* ... */ };
```

This class hierarchy is shown in the Fig. 13.6c.

Inheritance and Encapsulation

13.6.3

In this subsection we demonstrate that the inheritance may lead to significant violation of the encapsulation, which may cause problems in the implementation of derived classes. We start with an example.

Example 15 The particles interact in the detector in different ways. Some of the interaction represent events that are subject to our investigation and need to be logged in the result file and further processed. (This means to record the particle type, energy, coordinates of the interaction etc.) But the events may appear in groups, so the `ResultFile` class will contain the methods `LogEvent()` and `LogEventGroup()`. The latter will get the vector containing data of several events as an argument. Suppose that both these methods are polymorphic.

15

At some later stage of the program development, we find that it is necessary to be aware of the total count of recorded events. The actual implementation of the `ResultFile` class does not support this feature and we cannot change it, e.g., because it is part of some program library.

The solution seems to be easy – we derive a new class, `CountedResultFile`, based on the `ResultFile`. The implementation could be as follows:

```
class CountedResultFile: public ResultFile
{
public:
    virtual void LogEvent(Event *e)
    {
        ResultFile::LogEvent(e);
        count++;
    }
};
```

```

    }
    virtual void LogEventGroup(vector<Event*> eg)
    {
        ResultFile::LogEventGroup(eg);
        count += eg.size();
    }
private:
    int count;
};

```

The overridden methods simply call the base class methods to log the events and then increase the count of the recorded events.

It may happen that we find that the `LogEventGroup()` method increases the count of recorded events incorrectly: After the call

```

LogFile *clf = new CountedLogFile;
clf -> LogEventGroup(eg);    // (*)

```

the count value increases by twice the number of the events in `eg`.

The reason might be that the implementation of the `LogEventGroup()` method internally calls the `LogEvent()` method in a loop. This is what happens:

1. The `(*)` statement calls the `LogEventGroup()` method. This is a polymorphic method, so the `CountedResultFile::LogEventGroup()` method is called.
2. This method calls the base class `LogEventGroup()` method.
3. The base class method calls the `LogEvent()` method in a loop. *But because these methods are polymorphic, the method of the actual type, i.e., the `CountedResultFile::LogEvent()` method is called.*
4. This method calls the base class method to record the event and increases the count of events. After that it returns to the `CountedResultFile::LogEventGroup()` method. This method increases the event count once again.

To implement the derived class properly, we need to know that the `ResultFile::LogEventGroup()` method internally calls the `ResultFile::LogEvent()` method. *But this is an implementation detail, not the part of the contract of the methods of the `ResultFile` class.*

Solution

This problem may easily be avoided by using interfaces and object composition (cf. class diagram in Fig. 13.7); it is necessary to use another design of the `ResultFile` class, as well as another design of the `CountedResultFile` class.

First we design the `ResultFileInterface` interface as follows:

```
class ResultFileInterface {
```

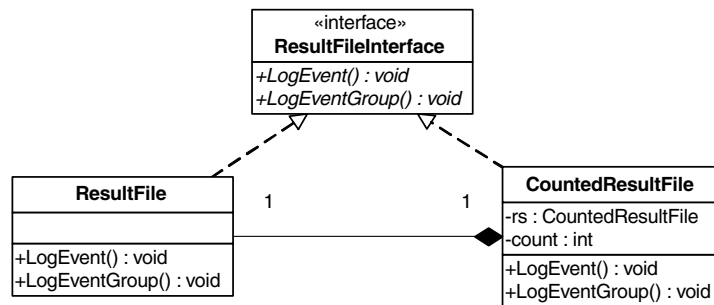



Figure 13.7. Class diagram of the correct design. Only interface methods and corresponding attributes are shown

```

public:
    virtual void LogEvent(Event *e) = 0;
    virtual void LogEventGroup(vector<Event*> eg) = 0;
};
  
```

The class `ResultFile` will implement this interface:

```

class ResultFile: public ResultFileInterface {
    // Implementation as before
};
  
```

Now, `CountedResultFile` may be designed as an independent class that implements the `ResultFileInterface` and uses the `ResultSet` as an attribute:

```

class CountedResultFile: public ResultFileInterface {
public:
    virtual void LogEvent(Event *e)
    {
        rs.LogEvent(e);
        count++;
    }
    virtual void LogEventGroup(vector<Event*> eg)
    {
        rs.LogEventGroup(eg);
        count += eg.size();
    }
private:
    int count;
    ResultSet rs;
};
  
```

The problem may not appear, because the `CountedResultFile` is not derived from the `ResultFile` now. Nevertheless, they may be treated polymorphically, i.e. instances of the `CountedResultFile` may be used instead of instances of the `ResultFile`, if they are used as instances of the `ResultFileInterface` interface.

13.7 Structure of the Object Oriented Program

We conclude this chapter by describing shortly the typical structure of the OOP program.

As we have seen in previous sections, an OOP program consists of objects that collaborate by messages. In this structure, one object must play the role of a *starting object*. This means that one of the methods of this object will be called as the program start. The starting object typically creates other objects in the program and manages their lifetime.

All the other objects represent various parts of the problem solved and are responsible for the associated resource management, computation, etc.

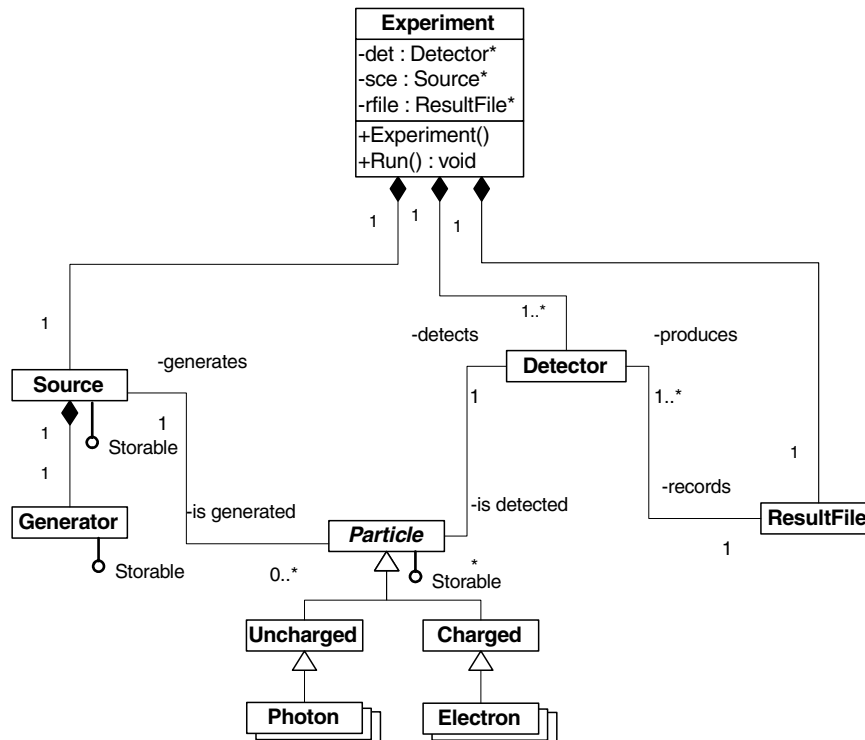


Figure 13.8. Basic structure of the simulation program

Example 16 Here we finish the Monte Carlo simulation of an experiment with particles. We have already mentioned the `Experiment` class covering the application as a whole. The only instance of this class in the program will be the starting object. The `Experiment` class will contain the public method `Run()` that will represent the run of the experiment.

As we are in C++, our program must have the `main()` function where the program starts. It will create the starting object and let it run:

```
int main() {                // Create the starting object
    Experiment().Run(); // and run it
}
```

The `Experiment` class could be defined as follows:

```
class Experiment{
public:
    Experiment();           // Constructor
    void Run();            // Run the experiment
private:
    Detector *det;         // Detector in this experiment
    Source *sce;           // Particle source
    ResultFile *rfile;     // Result file
};
```

The `Experiment::Experiment()` constructor reads the input data (e.g., cross sections describing the probabilities of the interactions) and creates and initializes the attributes (particle source, result file etc.).

The `Experiment::run()` methods does the work – it starts particle emission in the source using the appropriate method of the `Source` class, determines, whether the given particle hits the detector using the appropriate `Detector` method, records the results of the detection into the `ResultFile` instance and in the end processes the results using the appropriate `ResultFile` methods.

Class diagram of the program at the level we have given here is presented in Fig. 13.8.

Note that this is top-level design only. The extent of this chapter allows us to demonstrate only the beginning of the object oriented approach to the example.

Conclusion

Attempts to formalize the process of object oriented analysis and design have been made since the beginning of the OOP. A widely used approach is described in Booch (1993).

In object oriented design some common problems – or tasks – may appear in many different situations. Reusable solutions of these problems are called *Design Patterns*. A well known example of design pattern is Singleton – the class that may have at most one instance. Another example is Responsibility Chain; this design pattern solves the problem how to find the proper processing of varying data, even if the way of processing may dynamically change.

The idea of design patterns and the 23 most common design patterns in OOP are described in Gamma (1999).

References

- Booch, G.: *Object-Oriented Analysis and Design with Applications*. Addison-Wesley, New York (1993).
- Booch, G., Rumbaugh, J., Jacobson, I.: *The Unified Modeling Language User Guide*. Addison-Wesley, New York (1999).
- Gamma, E., Helm, R., Johnson R., Vlissides, J.: *Design Patterns*. Addison-Wesley, New York (1995).
- Liskov, B.: *Data Abstraction and Hierarchy*. SIGPLAN Notices, 23, 5 (May 1988).
- Martin, R. C.: *The Liskov Substitution Principle*. The C++ Report, (March 1996).
- Meyer, B.: *Object-Oriented Software Construction*. Prentice Hall, New York (1997); see also http://en.wikipedia.org/wiki/Eiffel_programming_language
- Stroustrup, B.: *The C++ Programming Language. Third edition*. Addison-Wesley, New York (1998).

Part III
Statistical Methodology



Model Selection

III.1

Yuedong Wang

1.1	<i>Introduction</i>	438
1.2	<i>Basic Concepts – Trade-Offs</i>	444
1.3	<i>AIC, BIC, C_p and Their Variations</i>	449
1.4	<i>Cross-Validation and Generalized Cross-Validation</i>	453
1.5	<i>Bayes Factor</i>	457
1.6	<i>Impact of Heteroscedasticity and Correlation</i>	460
1.7	<i>Discussion</i>	462

Introduction

The need for model selection arises when a data-based choice among competing models has to be made. For example, for fitting parametric regression (linear, non-linear and generalized linear) models with multiple independent variables, one needs to decide which variables to include in the model (Chapts. III.7, III.8 and III.12); for fitting non-parametric regression (spline, kernel, local polynomial) models, one needs to decide the amount of smoothing (Chapts. III.5 and III.10); for unsupervised learning, one needs to decide the number of clusters (Chapts. III.13 and III.16); and for tree-based regression and classification, one needs to decide the size of a tree (Chapt. III.14).

Model choice is an integral and critical part of data analysis, an activity which has become increasingly more important as the ever increasing computing power makes it possible to fit more realistic, flexible and complex models. There is a huge literature concerning this subject (Linhart and Zucchini, 1986; Miller, 2002; Burnham and Anderson, 2002; George, 2000) and we shall restrict this chapter to basic concepts and principles. We will illustrate these basics using a climate data, a simulation, and two regression models: parametric trigonometric regression and non-parametric periodic splines. We will discuss some commonly used model selection methods such as Akaike's AIC (Akaike, 1973), Schwarz's BIC (Schwarz, 1978), Mallows's C_p (Mallows, 1973), cross-validation (CV) (Stone, 1974), generalized cross-validation (GCV) (Craven and Wahba, 1979) and Bayes factors (Kass and Raftery, 1995). We do not intend to provide a comprehensive review. Readers may find additional model selection methods in the following chapters.

Let $\mathcal{M} = \{M_\lambda, \lambda \in \Lambda\}$ be a collection of candidate models from which one will select a model for the observed data. λ is the model index belonging to a set Λ which may be finite, countable or uncountable.

Variable selection in multiple regression is perhaps the most common form of model selection in practice. Consider the following linear model

$$y_i = \mathbf{x}_i^\top \boldsymbol{\beta} + \varepsilon_i, \quad i = 1, 2, \dots, n, \quad (1.1)$$

where \mathbf{x}_i are vectors of m independent variables, $\boldsymbol{\beta}$ is a m -vector of parameters, and ε_i are random errors. Often a large number of independent variables are investigated in the model (1.1) and it is likely that not all m variable are important. Statistical inferences can be carried out more efficiently with smaller models. The goal of the variable selection is to find a subset of these m independent variables which is optimal in a certain sense. In this case, Λ is the collection of all 2^m subsets and λ is any particular subset.

For illustration, we will use part of a climate data set downloaded from the Carbon Dioxide Information Analysis Center at <http://cdiac.ornl.gov/ftp/ndpo70>. The data consists of daily maximum temperatures and other climatological variables from 1062 stations across the contiguous United States. We choose daily maximum temperatures from the station in Charleston, South Carolina, which has the longest records from 1871 to 1997. We use records in the year 1990 as observations. Records

from other years provide information about the population. To avoid correlation (see Sect. 1.6) and simplify the presentation, we divided 365 days in 1990 into 73 five-day periods. The measurements on the third day in each period is selected as observations. Thus the data we use in our analyses is a subset consisting of every fifth day records in 1990 and the total number of observations $n = 73$. For simplicity, we transform the time variable t into the interval $[0, 1]$. The data is shown in the left panel of Fig. 1.1.

Our goal is to investigate how maximum temperature changes over time in a year. Consider a regression model

$$y_i = f(t_i) + \varepsilon_i, \quad t_i = i/n, \quad i = 1, \dots, n, \quad (1.2)$$

where y_i is the observed maximum temperature at time t_i in Fahrenheit, f is the mean temperature function and ε_i 's are random fluctuations in the year 1990. We assume that ε_i 's are independent and identically distributed with mean zero and variance σ^2 . Note that even though model (1.2) is very general, certain model assumptions (choices) have already been made implicitly. For example, the random fluctuations are assumed to be additive, independent and homogeneous. Violations of these assumptions such as independence may have severe impacts on model selection procedures (Sect. 1.6).

In the middle panel of Fig. 1.1, we plot observations on the same selected 73 days from 1871 to 1997. Assuming model (1.2) is appropriate for all years, the points represent 127 realizations from model (1.2). The averages reflect the true mean function f and the ranges reflect fluctuations. In the right panel, a smoothed version of the averages is shown, together with the observations in 1990. One may imagine that these observations were generated from the smoothed curve plus random errors. Our goal is to recover f from the noisy data. Before proceeding to estimation, one needs to decide a model space for the function f . Intuitively, a larger space provides greater potential to recover or approximate f . At the same time, a larger space makes model identification and estimation more difficult (Yang, 1999). Thus the greater potential provided by the larger space is more difficult to reach. One should use as much prior knowledge as possible to narrow down the

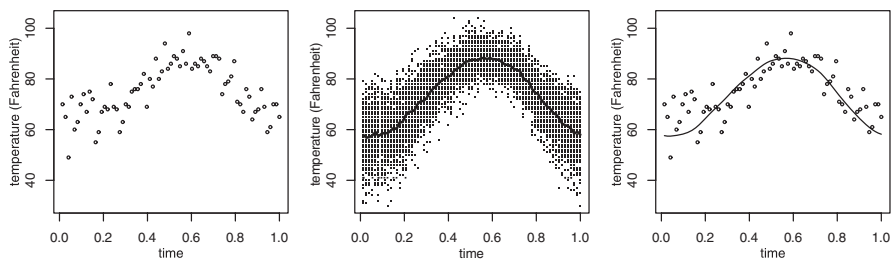


Figure 1.1. Left: 73 observations in the year 1990. Middle: observations on the same 73 days from 1871–1997. Averages are marked as the solid line. Right: 73 observations in the year 1990 (points) and a periodic spline fit (line) to the average temperatures in the middle panel

choice of model spaces. Since f represents mean maximum temperature in a year, we will assume that f is a periodic function.

Trigonometric Regression Model. It is a common practice to fit the periodic function f using a trigonometric model up to a certain frequency, say λ , where $0 \leq \lambda \leq K$ and $K = (n-1)/2 = 36$. Then the model space is

$$M_\lambda = \text{span} \left\{ 1, \sqrt{2} \sin 2\pi vt, \sqrt{2} \cos 2\pi vt, v = 1, \dots, \lambda \right\}. \quad (1.3)$$

The order λ is unknown in most applications. Thus one needs to select λ among $\Lambda = \{0, 1, \dots, K\}$ where $M_0 = \text{span}\{1\}$. For a fixed λ , we write the model M_λ as

$$y_i = \beta_1 + \sum_{v=1}^{\lambda} \left(\beta_{2v} \sqrt{2} \sin 2\pi vt + \beta_{2v+1} \sqrt{2} \cos 2\pi vt \right) + \varepsilon_i, \quad i = 1, \dots, n, \quad (1.4)$$

or in a matrix form

$$\mathbf{y} = X_\lambda \boldsymbol{\beta}_\lambda + \boldsymbol{\varepsilon},$$

where $\mathbf{y} = (y_1, \dots, y_n)^\top$,

$$X_\lambda = \begin{pmatrix} 1 & \sqrt{2} \sin 2\pi t_1 & \sqrt{2} \cos 2\pi t_1 & \cdots & \sqrt{2} \sin 2\pi \lambda t_1 & \sqrt{2} \cos 2\pi \lambda t_1 \\ 1 & \sqrt{2} \sin 2\pi t_2 & \sqrt{2} \cos 2\pi t_2 & \cdots & \sqrt{2} \sin 2\pi \lambda t_2 & \sqrt{2} \cos 2\pi \lambda t_2 \\ \vdots & \vdots & \vdots & \cdots & \vdots & \vdots \\ 1 & \sqrt{2} \sin 2\pi t_n & \sqrt{2} \cos 2\pi t_n & \cdots & \sqrt{2} \sin 2\pi \lambda t_n & \sqrt{2} \cos 2\pi \lambda t_n \end{pmatrix}$$

is the design matrix, $\boldsymbol{\beta}_\lambda = (\beta_1, \dots, \beta_{2\lambda+1})^\top$ and $\boldsymbol{\varepsilon} = (\varepsilon_1, \dots, \varepsilon_n)^\top$. The coefficients $\boldsymbol{\beta}_\lambda$ are estimated by minimizing the least squares (LS)

$$\min_{\boldsymbol{\beta}_\lambda} \left\{ \frac{1}{n} \sum_{i=1}^n \left(y_i - \beta_1 - \sum_{v=1}^{\lambda} \left(\beta_{2v} \sqrt{2} \sin 2\pi vt_i + \beta_{2v+1} \sqrt{2} \cos 2\pi vt_i \right) \right)^2 \right\} \quad (1.5)$$

Since design points are equally spaced, we have the following orthogonality relations:

$$\begin{aligned} \frac{2}{n} \sum_{i=1}^n \cos 2\pi vt_i \cos 2\pi \mu t_i &= \delta_{v,\mu}, \quad 1 \leq v, \mu \leq K, \\ \frac{2}{n} \sum_{i=1}^n \sin 2\pi vt_i \sin 2\pi \mu t_i &= \delta_{v,\mu}, \quad 1 \leq v, \mu \leq K, \\ \frac{2}{n} \sum_{i=1}^n \cos 2\pi vt_i \sin 2\pi \mu t_i &= 0, \quad 1 \leq v, \mu \leq K, \end{aligned} \quad (1.6)$$

where $\delta_{\nu,\mu}$ is the Kronecker delta. Thus columns of the design matrix are orthogonal. That is, $X_\lambda^\top X_\lambda = nI_{2\lambda+1}$ where $I_{2\lambda+1}$ is an identity matrix of size $2\lambda + 1$. Let X_K be the design matrix of the largest model M_K . Then X_K/\sqrt{n} is an orthonormal matrix. Define the discrete Fourier transformation $\tilde{y} = X_K^\top y/n$. The LS estimate of β_λ is $\hat{\beta}_\lambda = (X_\lambda^\top X_\lambda)^{-1} X_\lambda^\top y = X_\lambda^\top y/n = \tilde{y}_\lambda$, where \tilde{y}_λ consists of the first $2\lambda + 1$ elements of \tilde{y} . More explicitly,

$$\begin{aligned} \hat{\beta}_0 &= \frac{1}{n} \sum_{i=1}^n y_i = \tilde{y}_1, \\ \hat{\beta}_{2\nu} &= \frac{\sqrt{2}}{n} \sum_{i=1}^n y_i \sin 2\pi\nu t_i = \tilde{y}_{2\nu}, \quad 1 \leq \nu \leq \lambda, \\ \hat{\beta}_{2\nu+1} &= \frac{\sqrt{2}}{n} \sum_{i=1}^n y_i \cos 2\pi\nu t_i = \tilde{y}_{2\nu+1}, \quad 1 \leq \nu \leq \lambda. \end{aligned} \tag{1.7}$$

Let \hat{f}_λ be the estimate of f where the dependence on λ is expressed explicitly. Then the fits

$$\hat{f}_\lambda \triangleq (\hat{f}_\lambda(t_1), \dots, \hat{f}_\lambda(t_n))^\top = X_\lambda \hat{\beta}_\lambda = P(\lambda)y,$$

where

$$P(\lambda) = X_\lambda (X_\lambda^\top X_\lambda)^{-1} X_\lambda^\top = X_\lambda X_\lambda^\top / n \tag{1.8}$$

is the projection matrix. Note that $P(K) = I_n$. Thus model M_K interpolates the data.

Fits for several λ (labeled as k in strips) are shown in the top two rows of Fig. 1.2. Obviously as λ increases from zero to K , we have a family of models ranging from a constant to interpolation. A natural question is that which model (λ) gives the “best” estimate of f .

Periodic Spline. In addition to the periodicity, it is often reasonable to assume that f is a smooth function of $t \in [0, 1]$. Specifically, we assume the following infinite dimensional space (Wahba, 1990; Gu, 2002)

$$\begin{aligned} W_2(per) = \left\{ f : f \text{ and } f' \text{ are absolutely continuous,} \right. \\ \left. f(0) = f(1), f'(0) = f'(1), \int_0^1 (f''(t))^2 dt < \infty \right\} \end{aligned} \tag{1.9}$$

as the model space for f . A smoothing spline estimate of f is the minimizer of the following penalized LS (Wahba, 1990; Gu, 2002)

$$\min_{f \in W_2(per)} \left\{ \frac{1}{n} \sum_{i=1}^n (y_i - f(t_i))^2 + \lambda \int_0^1 (f''(t))^2 dt \right\}, \tag{1.10}$$

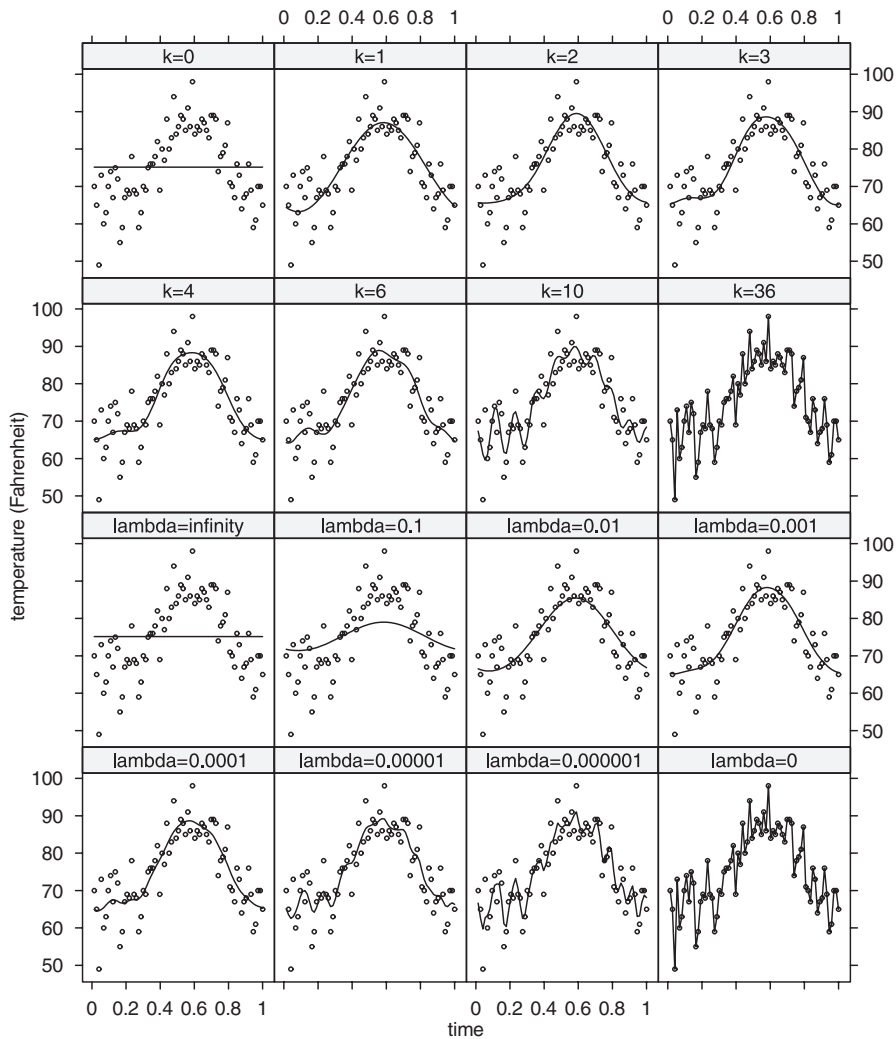


Figure 1.2. Circles are observations. Lines are the estimated functions. Top two rows are fits from trigonometric models with frequencies indicated in the strips (we used k instead of λ for distinction). Bottom two rows are fits from the periodic spline with smoothing parameters indicated in the strips

where the first part (LS) measures the goodness-of-fit, the second part is a penalty to the roughness of the estimate, and λ ($0 \leq \lambda \leq \infty$) is the so called *smoothing parameter* which controls the trade-off between the goodness-of-fit and the roughness. When $\lambda = 0$, there is no penalty to the roughness of the estimate and the spline estimate interpolates the data. When $\lambda = \infty$, the spline estimate is forced to be a constant. As λ varies from zero to infinity, we have a family of models ranging

from interpolation to a constant parametric model. Thus λ can be considered as a model index in $\Lambda = [0, \infty]$. Fits with several choices of λ are shown in the bottom two rows of Fig. 1.2.

The exact solution to (1.10) can be found in Wahba (1990). To simplify the argument, as in Wahba (1990), we consider the following approximation of the original problem

$$\min_{f \in M_K} \left\{ \frac{1}{n} \sum_{i=1}^n (y_i - f(t_i))^2 + \lambda \int_0^1 (f''(t))^2 dt \right\}, \tag{1.11}$$

where $W_2(per)$ in (1.10) is replaced by M_K which is defined in (1.3) with $\lambda = K$. The following discussions hold true for the exact solution in the infinite dimensional spaces (Wahba, 1990; Gu, 2002). The approximation makes the following argument transparent and provides insights into smoothing.

Let

$$\widehat{f}_\lambda(t) = \widehat{\alpha}_1 + \sum_{\nu=1}^K \left(\widehat{\alpha}_{2\nu} \sqrt{2} \sin 2\pi\nu t + \widehat{\alpha}_{2\nu+1} \sqrt{2} \cos 2\pi\nu t \right)$$

be the solution to (1.11). Then $\widehat{\mathbf{f}}_\lambda \triangleq (\widehat{f}_\lambda(t_1), \dots, \widehat{f}_\lambda(t_n))^\top = X_K \widehat{\boldsymbol{\alpha}}$, where $\widehat{\boldsymbol{\alpha}} = (\widehat{\alpha}_1, \dots, \widehat{\alpha}_{2K+1})^\top$. The LS

$$\frac{1}{n} \|\mathbf{y} - \widehat{\mathbf{f}}_\lambda\|^2 = \frac{1}{n} \left\| \frac{1}{\sqrt{n}} X_K^\top (\mathbf{y} - \widehat{\mathbf{f}}_\lambda) \right\|^2 = \left\| \frac{1}{n} X_K^\top \mathbf{y} - \frac{1}{n} X_K^\top X_K \widehat{\boldsymbol{\alpha}} \right\|^2 = \|\widehat{\mathbf{y}} - \widehat{\boldsymbol{\alpha}}\|^2.$$

Thus (1.11) reduces to the following ridge regression problem

$$(\widehat{\alpha}_1 - \widehat{y}_1)^2 + \sum_{\nu=1}^K \left((\widehat{\alpha}_{2\nu} - \widehat{y}_{2\nu})^2 + (\widehat{\alpha}_{2\nu+1} - \widehat{y}_{2\nu+1})^2 \right) + \lambda \sum_{\nu=1}^K (2\pi\nu)^4 (\widehat{\alpha}_{2\nu}^2 + \widehat{\alpha}_{2\nu+1}^2). \tag{1.12}$$

The solutions to (1.12) are

$$\begin{aligned} \widehat{\alpha}_1 &= \widehat{y}_1, \\ \widehat{\alpha}_{2\nu} &= \widehat{y}_{2\nu} / (1 + \lambda(2\pi\nu)^4), \quad \nu = 1, \dots, K, \\ \widehat{\alpha}_{2\nu+1} &= \widehat{y}_{2\nu+1} / (1 + \lambda(2\pi\nu)^4), \quad \nu = 1, \dots, K. \end{aligned} \tag{1.13}$$

Thus the periodic spline with equally spaced design points is essentially a low-pass filter: components at frequency ν are down-weighted by a factor of $1 + \lambda(2\pi\nu)^4$. The right panel of Fig. 1.3 shows how λ controls the nature of the filter: more high frequencies are filtered out as λ increases. It is clear from (1.7) and (1.13) that selecting an order for the trigonometric model may be viewed as hard thresholding and selecting the smoothing parameter for the periodic spline may be viewed as soft thresholding.

Let $D = \text{diag}(1, 1/(1+\lambda(2\pi)^4), 1/(1+\lambda(2\pi)^4), \dots, 1/(1+\lambda(2\pi K)^4), 1/(1+\lambda(2\pi K)^4))$. Then $\hat{\alpha} = D\tilde{y}$, and the fit

$$\hat{f}_\lambda = X_K \hat{\alpha} = \frac{1}{n} X_K D X_K^\top \mathbf{y} = A(\lambda) \mathbf{y},$$

where

$$A(\lambda) = X_K D X_K^\top / n \quad (1.14)$$

is the hat (smoother) matrix.

We choose the trigonometric regression and periodic spline models for illustration because of their simple model indexing: the first has a finite set of consecutive integers $\Lambda = \{0, 1, \dots, K\}$ and the second has a continuous interval $\Lambda = [0, \infty]$.

This chapter is organized as follows. In Sect. 1.2, we discuss the trade-offs between the goodness-of-fit and model complexity, and the trade-offs between bias and variance. We also introduce mean squared error as a target criterion. In Sect. 1.3, we introduce some commonly used model selection methods: AIC, BIC, C_p , AIC_c and a data-adaptive choice of the penalty. In Sect. 1.4, we discuss the cross-validation and the generalized cross-validation methods. In Sect. 1.5, we discuss Bayes factor and its approximations. In Sect. 1.6, we illustrate potential effects of heteroscedasticity and correlation on model selection methods. The chapter ends with some general comments in Sect. 1.7.

1.2 Basic Concepts – Trade-Offs

We illustrate in this section that model selection boils down to compromises between different aspects of a model. Occam's razor has been the guiding principle for the compromises: the model that *fits observations sufficiently well in the least complex way* should be preferred. Formalization of this principle is, however, nontrivial.

To be precise on *fits observations sufficiently well*, one needs a quantity that measures how well a model fits the data. This quantity is often called the *goodness-of-fit* (GOF). It usually is the criterion used for estimation, after deciding on a model. For example, we have used the LS as a measure of the GOF for regression models in Sect. 1.1. Other GOF measures include likelihood for density estimation problems and classification error for pattern recognition problems.

To be precise on *the least complex way*, one needs a quantity that measures the complexity of a model. For a parametric model, a common measure of model complexity is the number of parameters in the model, often called the *degrees of freedom* (df). For a non-parametric regression model like the periodic spline, $trA(\lambda)$, a direct extension from its parametric version, is often used as a measure of model complexity (Hastie and Tibshirani, 1990). $trA(\lambda)$ will also be referred to as the degrees of freedom. The middle panel of Fig. 1.3 depicts how $trA(\lambda)$

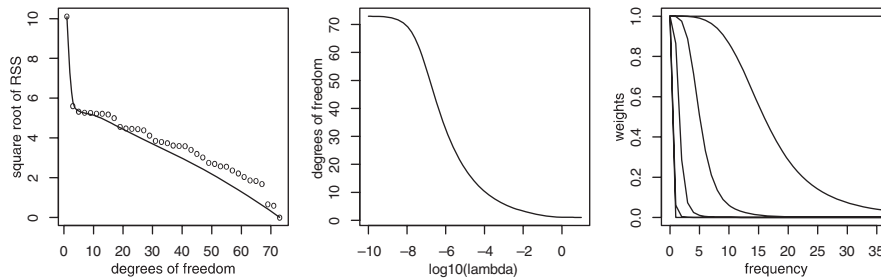


Figure 1.3. *Left*: square root of RSS from the trigonometric model (*circles*) and periodic spline (*line*) plotted against the degrees of freedom. *Middle*: degrees of freedom of periodic spline plotted against the smoothing parameter on the logarithm base 10 scale. *Right*: weights of the periodic spline filter, $1/(1 + \lambda(2\pi\nu)^4)$, plotted as a function of frequency ν . Six curves from top down corresponds to six different λ : 0, 10^{-8} , 10^{-6} , 10^{-4} , 10^{-2} and ∞

for the periodic spline depends on the smoothing parameter λ . Since there is an one-to-one correspondence between λ and $trA(\lambda)$, both of them are used as model index (Hastie and Tibshirani, 1990). See Gu (1998) for discussions on some subtle issues concerning model index for smoothing spline models. For some complicated models such as tree-based regression, there may not be an obvious measure of model complexity (Ye, 1998). In these cases the generalized degrees of freedom defined in Ye (1998) may be used. Section 1.3 contains more details on the generalized degrees of freedom.

To illustrate the interplay between the GOF and model complexity, we fit trigonometric regression models from the smallest model with $\lambda = 0$ to the largest model with $\lambda = K$. The square root of residual sum of squares (RSS) are plotted against the degrees of freedom ($= 2\lambda + 1$) as circles in the left panel of Fig. 1.3. Similarly, we fit the periodic spline with a wide range of values for the smoothing parameter λ . Again, we plot the square root of RSS against the degrees of freedom ($= trA(\lambda)$) as the solid line in the left panel of Fig. 1.3. Obviously, RSS decreases to zero (interpolation) as the degrees of freedom increases to n . RSS keeps declining almost linearly after the initial big drop. It is quite clear that the constant model does not fit data well. But it is unclear which model *fits observations sufficiently well*.

The previous example shows that the GOF and complexity are two opposite aspects of a model: the approximation error decreases as the model complexity increases. On the other hand, the Occam's razor suggests that simple models should be preferred to more complicated ones, other things being equal. Our goal is to find the "best" model that strikes a balance between these two conflicting aspects. To make the word "best" meaningful, one needs a target criterion which quantifies a model's performance. It is clear that the GOF cannot be used as the target because it will lead to the most complex model. Even though there is no universally accepted measure, some criteria are widely accepted and used in practice. We now discuss one of them which is commonly used for regression models.

Let \widehat{f}_λ be an estimate of the function in model (1.2) based on the model space M_λ . Let $\mathbf{f} = (f(t_1), \dots, f(t_n))^\top$ and $\widehat{\mathbf{f}}_\lambda = (\widehat{f}_\lambda(t_1), \dots, \widehat{f}_\lambda(t_n))^\top$. Define the mean squared error (MSE) by

$$\text{MSE}(\lambda) = \mathbb{E} \left(\frac{1}{n} \|\widehat{\mathbf{f}}_\lambda - \mathbf{f}\|^2 \right).$$

We want the estimate \widehat{f}_λ to be as close to the true function f as possible. Obviously MSE is the expectation of the Euclidean distance between the estimates and the true values. L_2 distance between \widehat{f}_λ and f may also be used. MSE can be decomposed into two components:

$$\begin{aligned} \text{MSE}(\lambda) &= \frac{1}{n} \mathbb{E} \|(\mathbb{E}\widehat{\mathbf{f}}_\lambda - \mathbf{f}) + (\widehat{\mathbf{f}}_\lambda - \mathbb{E}\widehat{\mathbf{f}}_\lambda)\|^2 \\ &= \frac{1}{n} \mathbb{E} \|\mathbb{E}\widehat{\mathbf{f}}_\lambda - \mathbf{f}\|^2 + \frac{2}{n} \mathbb{E}(\mathbb{E}\widehat{\mathbf{f}}_\lambda - \mathbf{f})^\top (\widehat{\mathbf{f}}_\lambda - \mathbb{E}\widehat{\mathbf{f}}_\lambda) + \frac{1}{n} \mathbb{E} \|\widehat{\mathbf{f}}_\lambda - \mathbb{E}\widehat{\mathbf{f}}_\lambda\|^2 \\ &= \frac{1}{n} \|\mathbb{E}\widehat{\mathbf{f}}_\lambda - \mathbf{f}\|^2 + \frac{1}{n} \mathbb{E} \|\widehat{\mathbf{f}}_\lambda - \mathbb{E}\widehat{\mathbf{f}}_\lambda\|^2 \\ &\triangleq \text{Bias}^2 + \text{Variance}. \end{aligned} \quad (1.15)$$

The Bias² measures how well the model M_λ approximates the true function f , and the Variance measures how well the function can be estimated in M_λ . Usually larger model space leads to smaller Bias² but larger Variance. Thus, the MSE represents a trade-off between Bias² and Variance.

Another closely related target criterion is the average predictive squared error (PSE)

$$\text{PSE}(\lambda) = \mathbb{E} \left(\frac{1}{n} \|\mathbf{y}^+ - \widehat{\mathbf{f}}_\lambda\|^2 \right), \quad (1.16)$$

where $\mathbf{y}^+ = \mathbf{f} + \boldsymbol{\varepsilon}^+$ are new observations at the same design points, $\boldsymbol{\varepsilon}^+ = (\varepsilon_1^+, \dots, \varepsilon_n^+)^\top$ are independent of $\boldsymbol{\varepsilon}$, and ε_i^+ 's are independent and identically distributed with mean zero and variance σ^2 . PSE measures the performance of a model's prediction for new observations. We have

$$\text{PSE}(\lambda) = \mathbb{E} \left(\frac{1}{n} \|(\mathbf{y}^+ - \mathbf{f}) + (\mathbf{f} - \widehat{\mathbf{f}}_\lambda)\|^2 \right) = \sigma^2 + \text{MSE}(\lambda).$$

Thus PSE differs from MSE only by a constant σ^2 . When justifying some criteria including the C_p in Sect. 1.3, we will ignore a constant σ^2 . Thus the targets of these criteria are really PSE rather than MSE.

To illustrate the bias-variance trade-off, we now calculate MSE for the trigonometric regression and periodic spline models. For notational simplicity, we assume that $f \in M_K$:

$$f(t) = \alpha_1 + \sum_{v=1}^K \left(\alpha_{2v} \sqrt{2} \sin 2\pi vt + \alpha_{2v+1} \sqrt{2} \cos 2\pi vt \right). \quad (1.17)$$

Then $\mathbf{f} = X_K \boldsymbol{\alpha}$ where $\boldsymbol{\alpha} = (\alpha_1, \dots, \alpha_n)^\top$. From the orthogonality relations (1.6), it is easy to check that $\boldsymbol{\alpha} = X_K^\top \mathbf{f}/n$, the discrete Fourier transformation of \mathbf{f} .

Bias-Variance Trade-Off for the Trigonometric Regression. X_λ consists of the first $2\lambda + 1$ columns of the orthogonal matrix X_K . Thus $X_\lambda^\top X_K = (nI_{2\lambda+1}, \mathbf{0})$. $E(\widehat{\boldsymbol{\beta}}_\lambda) = X_\lambda^\top X_K \boldsymbol{\alpha}/n = \boldsymbol{\alpha}_\lambda$, where $\boldsymbol{\alpha}_\lambda$ consists of the first $2\lambda + 1$ elements in $\boldsymbol{\alpha}$. Thus $\widehat{\boldsymbol{\beta}}_\lambda$ is unbiased. Furthermore,

$$\begin{aligned} \text{Bias}^2 &= \frac{1}{n} \|(I_n - P(\lambda))\mathbf{f}\|^2 \\ &= \frac{1}{n} \left\| \frac{1}{\sqrt{n}} X_K^\top \left(I_n - \frac{1}{n} X_\lambda X_\lambda^\top \right) X_K \boldsymbol{\alpha} \right\|^2 \\ &= \frac{1}{n^2} \left\| \left(nI_n - \begin{pmatrix} nI_{2\lambda+1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{pmatrix} \right) \boldsymbol{\alpha} \right\|^2 \\ &= \sum_{v=\lambda+1}^K (\alpha_{2v}^2 + \alpha_{2v+1}^2), \\ \text{Variance} &= \frac{1}{n} E\|P(\lambda)(\mathbf{y} - \mathbf{f})\|^2 = \frac{\sigma^2}{n} \text{tr}P^2(\lambda) = \frac{\sigma^2}{n} (2\lambda + 1). \end{aligned}$$

Thus

$$\text{MSE}(\lambda) = \sum_{v=\lambda+1}^K (\alpha_{2v}^2 + \alpha_{2v+1}^2) + \frac{1}{n} \sigma^2 (2\lambda + 1).$$

Remarks

1. Adding the λ th frequency terms into the model space reduces the Bias² by the amount of $(\alpha_{2\lambda}^2 + \alpha_{2\lambda+1}^2)$ and increases the Variance by $2\sigma^2/n$.
2. The optimal model based on the MSE may not be the true model when $\sigma^2 > 0$.
3. Assuming $\alpha_{2\lambda}^2 + \alpha_{2\lambda+1}^2$ decreases with increasing λ , one should keep adding frequencies until

$$\alpha_{2\lambda}^2 + \alpha_{2\lambda+1}^2 \leq 2\sigma^2/n. \tag{1.18}$$

4. Bias² does not depend on the sample size n and the Variance is inversely proportional to n . Thus as n increases, more frequencies should be included.

Bias-Variance Trade-Off for Periodic Spline. For the approximate periodic spline estimator, it is easy to check that $E(\widehat{\mathbf{y}}) = \boldsymbol{\alpha}$, $\text{Var}(\widehat{\mathbf{y}}) = \sigma^2 I/n$, $E(\widehat{\boldsymbol{\alpha}}) = D\boldsymbol{\alpha}$, $\text{Var}(\widehat{\boldsymbol{\alpha}}) = \sigma^2 D^2$, $E(\widehat{f}_\lambda) = X_K D \boldsymbol{\alpha}$, and $\text{Var}(\widehat{f}_\lambda) = \sigma^2 X_K D^2 X_K^\top/n$. Thus all coefficients are shrunk to zero except $\widehat{\alpha}_1$ which is unbiased. The amount of shrinkage is controlled by the

smoothing parameter λ . It is straightforward to calculate the Bias² and Variance in (1.15).

$$\begin{aligned} \text{Bias}^2 &= \frac{1}{n} \|X_K \boldsymbol{\alpha} - X_K D \boldsymbol{\alpha}\|^2 = \frac{1}{n} \left\| \frac{1}{\sqrt{n}} X_K^\top (X_K \boldsymbol{\alpha} - X_K D \boldsymbol{\alpha}) \right\|^2 \\ &= \|(I - D) \boldsymbol{\alpha}\|^2 = \sum_{\nu=1}^K \left(\frac{\lambda(2\pi\nu)^4}{1 + \lambda(2\pi\nu)^4} \right)^2 (\alpha_{2\nu}^2 + \alpha_{2\nu+1}^2), \end{aligned}$$

$$\text{Variance} = \frac{1}{n^2} \sigma^2 \text{tr}(X_K D^2 X_K^\top) = \frac{\sigma^2}{n} \text{tr}(D^2) = \frac{\sigma^2}{n} \left(1 + 2 \sum_{\nu=1}^K \left(\frac{1}{1 + \lambda(2\pi\nu)^4} \right)^2 \right).$$

Thus

$$\text{MSE} = \sum_{\nu=1}^K \left(\frac{\lambda(2\pi\nu)^4}{1 + \lambda(2\pi\nu)^4} \right)^2 (\alpha_{2\nu}^2 + \alpha_{2\nu+1}^2) + \frac{\sigma^2}{n} \left(1 + 2 \sum_{\nu=1}^K \left(\frac{1}{1 + \lambda(2\pi\nu)^4} \right)^2 \right).$$

It is easy to see that as λ increases from zero to infinity, the Bias² increases from zero to $\sum_{\nu=1}^K (\alpha_{2\nu}^2 + \alpha_{2\nu+1}^2)$ and the Variance decreases from σ^2 to σ^2/n .

To calculate MSE, one needs to know the true function. We use the following simulation for illustration. We generate responses from model (1.2) with $f(t) = \sin(4\pi t^2)$ and $\sigma = 0.5$. The same design points in the climate data is used: $t_i = i/n$, $i = 1, \dots, n$ and $n = 73$. The true function and responses are shown in the left panel of Fig. 1.4. We compute $b_\nu = \log(a_{2\nu}^2 + a_{2\nu+1}^2)$, $\nu = 1, \dots, K$. b_ν represents the contribution from frequency ν . In the right panel of Fig. 1.4, we plot b_ν against frequency ν with the threshold, $\log(2\sigma^2/n)$, marked as the dashed line. Except for $\nu = 1$, b_ν decreases as ν increases. Values of b_ν are above the threshold for the first four frequencies. Thus the optimal choice is $\nu = 4$.

Bias², Variance and MSE are plotted against frequency ($\log_{10}(\lambda)$) for trigonometric regression (periodic spline) in the left (right) panel of Fig. 1.5. Obviously, as the frequency (λ) increases (decreases), the Bias² decreases and the Variance increases.

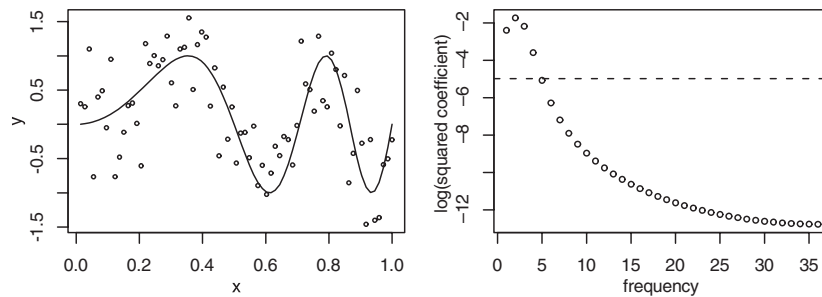


Figure 1.4. Left: true function for the simulation (line) and observations (circle). Right: plots of b_ν , $\nu = 1, \dots, K$, as circles and the threshold, $\log(2\sigma^2/n)$, as the dashed line

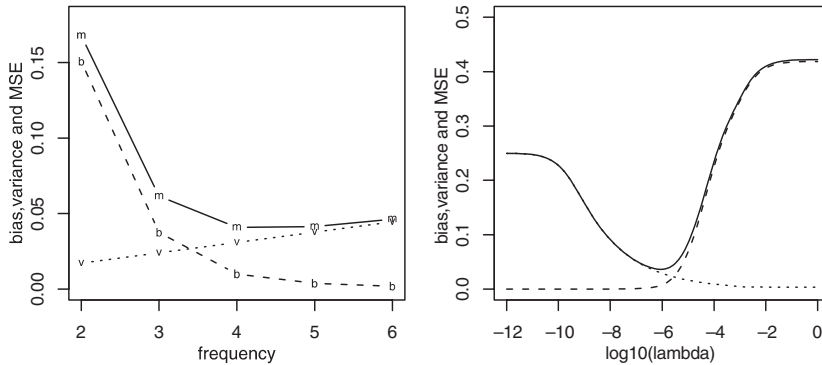


Figure 1.5. Bias² (dashed lines), Variance (dotted lines) and MSE (solid lines) for trigonometric regression on the left and periodic spline on the right

The MSE represents a balance between Bias² and Variance. For the trigonometric regression, the minimum of the MSE is reached at $\nu = 4$, as expected.

After deciding on a target criterion such as the MSE, ideally one would select the model to minimize this criterion. This is, however, not practical because the criterion usually involves some unknown quantities. For example, MSE depends on the unknown true function f which one wants to estimate in the first place. Thus one needs to estimate this criterion from the data and then minimize the estimated criterion. We discuss unbiased and cross-validation estimates of MSE in Sects 1.3 and 1.4 respectively.

AIC, BIC, C_p and Their Variations

Let $GOF(M_\lambda)$ and $|M_\lambda|$ be measures of the GOF and complexity for model M_λ . A direct compromise between these two conflicting quantities is

$$GOF(M_\lambda) + \xi|M_\lambda|, \tag{1.19}$$

where the parameter ξ controls how this compromise should be achieved. Note that the penalized LS (1.10) may be considered as a special case with the LS regarded as a measure of GOF and the squared integral regarded as a measure of complexity.

To be concrete, let us consider the regression model (1.2). For a fixed λ , the estimates are linear, $\hat{f}_\lambda = H(\lambda)y$, where $H(\lambda) = P(\lambda)$ for the trigonometric model and $H(\lambda) = A(\lambda)$ for the periodic spline. Suppose that LS is used as the measure of GOF and $|M_\lambda| = tr(H(\lambda))$. Let us first consider the case when the error variance σ^2 is known. Then the criterion (1.19) can be re-expressed as

$$U(\lambda, \theta) = \frac{1}{n} \|y - \hat{f}_\lambda\|^2 + \frac{\theta}{n} \sigma^2 trH(\lambda). \tag{1.20}$$

(1.20) is known as the final prediction error (FPE) criterion (Akaike, 1970). Many existing model selection methods corresponds to special choices of θ : $\theta = 2$ for Akaike's AIC and Mallows's C_p , and $\theta = \log n$ for Schwarz's BIC. The C_p method is also called the *unbiased risk method* (UBR) in smoothing spline literature (Wahba, 1990). The following simple argument provides the motivation behind the C_p (UBR) method.

$$\begin{aligned} \mathbb{E} \left(\frac{1}{n} \|\mathbf{y} - \widehat{\mathbf{f}}_\lambda\|^2 \right) &= \mathbb{E} \left(\frac{1}{n} \|\mathbf{y} - \mathbf{f}\|^2 + \frac{2}{n} (\mathbf{y} - \mathbf{f})^\top (\mathbf{f} - \widehat{\mathbf{f}}_\lambda) + \frac{1}{n} \|\mathbf{f} - \widehat{\mathbf{f}}_\lambda\|^2 \right) \\ &= \sigma^2 - \frac{2}{n} \sigma^2 \text{tr} H(\lambda) + \text{MSE}(\lambda). \end{aligned} \quad (1.21)$$

Therefore, ignoring the constant σ^2 , $U(\lambda, 2)$ is an unbiased estimate of $\text{MSE}(\lambda)$.

Other choices of θ were motivated from different principles: AIC is an estimate of the expected Kullback-Leibler discrepancy where the second term in (1.20) is considered as a bias correction (Burnham and Anderson, 2002) and BIC is an asymptotic Bayes factor (Sect. 1.5). Since each method was derived with different motivations, it is not surprising that they have quite different theoretical properties (Shao, 1997). θ in (1.20) can be considered as a penalty to the model complexity. A larger penalty (θ) leads to a simpler model. As a result, AIC and C_p perform well for "complex" true models and poorly for "simple" true models, while BIC does just the opposite. In practice the nature of the true model, "simple" or "complex", is never known. Thus a data driven choice of model complexity penalty θ would be desirable. Several methods have been proposed to estimate θ (Rao and Wu, 1989; Rao and Tibshirani, 1997; Bai et al., 1999; Rao, 1999; Shen and Ye, 2002). We now discuss Shen and Ye (2002)'s method based on the generalized degrees of freedom. We will discuss the cross-validation method (Rao and Tibshirani, 1997) in the next section.

Now consider both λ and θ in (1.20) as unknown parameters. Denote $\widehat{\lambda}(\theta)$ as the selected model index based on (1.20) for a fixed θ , and $\widehat{\mathbf{f}}_{\widehat{\lambda}(\theta)}$ as the estimate based on the selected model. The dependence on θ is made explicit. We now want to find θ which minimizes the MSE

$$\text{MSE}(\theta) = \mathbb{E} \left(\frac{1}{n} \|\widehat{\mathbf{f}}_{\widehat{\lambda}(\theta)} - \mathbf{f}\|^2 \right).$$

Again, we need to estimate $\text{MSE}(\theta)$. As (1.21), we have

$$\begin{aligned} \mathbb{E} \left(\frac{1}{n} \|\mathbf{y} - \widehat{\mathbf{f}}_{\widehat{\lambda}(\theta)}\|^2 \right) &= \mathbb{E} \left(\frac{1}{n} \|\mathbf{y} - \mathbf{f}\|^2 + \frac{2}{n} (\mathbf{y} - \mathbf{f})^\top (\mathbf{f} - \widehat{\mathbf{f}}_{\widehat{\lambda}(\theta)}) + \frac{1}{n} \|\mathbf{f} - \widehat{\mathbf{f}}_{\widehat{\lambda}(\theta)}\|^2 \right) \\ &= \sigma^2 - \frac{2}{n} \sigma^2 g_0(\theta) + \text{MSE}(\theta), \end{aligned}$$

where

$$g_0(\theta) = \mathbb{E} \mathbf{E}^\top (\widehat{\mathbf{f}}_{\widehat{\lambda}(\theta)} - \mathbf{f}) / \sigma^2$$

is the *generalized degrees of freedom* (gdf) defined in Ye (1998). Thus, ignoring a constant σ^2 ,

$$G(\theta) = \frac{1}{n} \|\mathbf{y} - \widehat{\mathbf{f}}_{\widehat{\lambda}(\theta)}\|^2 + \frac{2}{n} \sigma^2 g_0(\theta) \tag{1.22}$$

is an unbiased estimate of $\text{MSE}(\theta)$. More rigorous justification can be found in Shen and Ye (2002). Note that $\widehat{\lambda}(\theta)$ depends on \mathbf{y} . Thus $\widehat{\mathbf{f}}_{\widehat{\lambda}(\theta)} = H(\widehat{\lambda}(\theta))\mathbf{y}$ is a non-linear estimator. Usually $g_0(\theta) \neq \text{tr}H(\widehat{\lambda}(\theta))$. We need to estimate the gdf $g_0(\theta)$. It can be shown that (Ye, 1998)

$$\widehat{g}_0(\theta) = \int \boldsymbol{\delta}^\top \widehat{\mathbf{f}}_{\widehat{\lambda}(\theta)}(\mathbf{y} + \boldsymbol{\delta}) \phi_\tau(\boldsymbol{\delta}) d\boldsymbol{\delta}$$

is an approximately unbiased estimate of $g_0(\theta)$, where $\boldsymbol{\delta} \sim N(\mathbf{0}, \tau^2 I)$, $\phi_\tau(\boldsymbol{\delta})$ is the n -dimensional density of $N(\mathbf{0}, \tau^2 I)$, and $\widehat{\mathbf{f}}_{\widehat{\lambda}(\theta)}(\mathbf{y} + \boldsymbol{\delta})$ is the fit to the perturbed data $\mathbf{y} + \boldsymbol{\delta}$. Ye (1998) suggested the following Monte Carlo approach to compute $\widehat{g}_0(\theta)$: for a fixed θ ,

1. draw a n -dimensional sample $\boldsymbol{\delta} \sim N(\mathbf{0}, \tau^2 I)$, use the perturbed sample $\mathbf{y} + \boldsymbol{\delta}$ to select a model based on (1.20) with fixed θ and calculate the fits $\widehat{\mathbf{f}}_{\widehat{\lambda}(\theta)}(\mathbf{y} + \boldsymbol{\delta})$.
2. Repeating above step T times, one has $\boldsymbol{\delta}_t = (\delta_{t1} \cdots, \delta_{tn})^\top$ and $\widehat{\mathbf{f}}_{\widehat{\lambda}(\theta)}(\mathbf{y} + \boldsymbol{\delta}_t) \triangleq (\widehat{f}_{t1}, \cdots, \widehat{f}_{tn})^\top, t = 1, \cdots, T$.
3. For a fixed $i, i = 1, \cdots, n$, calculate the regression slope \widehat{h}_i of the following linear model

$$\widehat{f}_{ti} = \mu + \widehat{h}_i \delta_{ti}, \quad t = 1, \cdots, T.$$

4. Estimate $g_0(\theta)$ by $\sum_{i=1}^n \widehat{h}_i$.

$\tau \in [0.5\sigma, \sigma]$ is generally a good choice and the results are usually insensitive to the choice of τ when it is in this region (Ye, 1998). A data-driven choice of θ is the minimum of (1.22) with $g_0(\theta)$ replaced by its estimate $\sum_{i=1}^n \widehat{h}_i$ (Shen and Ye, 2002).

When σ^2 is unknown, one may replace σ^2 in (1.20) and (1.22) by a consistent estimate. Many estimators were proposed in literature (Rice, 1984; Gasser et al., 1986; Dette et al., 1998; Hall et al., 1990; Donoho and Johnston, 1994). The Rice's estimator is one of the simplest. For model (1.2), Rice (1984) proposed to estimate σ^2 by

$$\widehat{\sigma}^2 = \frac{1}{2(n-1)} \sum_{i=2}^n (y_i - y_{i-1})^2.$$

In the remaining of this chapter, σ^2 is replaced by $\widehat{\sigma}^2$ whenever necessary.

Another option, assuming the distribution of y_i 's is known, is to replace $\text{GOF}(M_\lambda)$ in (1.19) by $-2 \log(\text{maximum likelihood})$. For the regression models with Gaussian random errors, this leads to

$$n \log(\|\mathbf{y} - \widehat{\mathbf{f}}_\lambda\|^2) + \theta \text{tr}H(\lambda). \tag{1.23}$$

Again, $\theta = 2$ and $\theta = \log n$ correspond to AIC and BIC criteria respectively. The same data-driven procedure discussed above may also be used to select θ .

Derived from asymptotic argument, the AIC method may lead to over-fitting for small samples (Burnham and Anderson, 2002; Hurvich and Tsai, 1989). The following AIC_c criterion modifies (1.23) with a second order bias adjustment (Hurvich and Tsai, 1989)

$$AIC_c = n \log(\|y - \hat{f}_\lambda\|^2) + 2trH(\lambda) \frac{n}{n - trH(\lambda) - 1} .$$

AIC_c should be used when the ratio between n and the number of parameters in the largest candidate model is small, say less than 40 (Burnham and Anderson, 2002). In our trigonometric model, the highest dimension may reach n . Thus we will use AIC_c in our computations.

Now consider the trigonometric model. It is easy to check that criterion (1.20) reduces to

$$\sum_{\nu=\lambda+1}^K (\hat{y}_{2\nu}^2 + \hat{y}_{2\nu+1}^2) + \frac{\theta}{n} \sigma^2 (2\lambda + 1) .$$

Thus adding the λ th frequency reduces RSS by $\hat{y}_{2\lambda}^2 + \hat{y}_{2\lambda+1}^2$ and increases the complexity part by $2\theta\sigma^2/n$. When $\hat{y}_{2\lambda}^2 + \hat{y}_{2\lambda+1}^2$ decreases with increasing λ , one should keep adding frequencies until $\hat{y}_{2\lambda}^2 + \hat{y}_{2\lambda+1}^2 \leq 2\theta\sigma^2/n$. It is not difficult to see that the C_p criterion corresponds to applying rule (1.18) with $\alpha_{2\lambda}^2 + \alpha_{2\lambda+1}^2$ replaced by its unbiased estimate $\hat{y}_{2\lambda}^2 + \hat{y}_{2\lambda+1}^2 - 2\sigma^2/n$. Other data-based thresholding can be found in Donoho and Johnston (1994), Beran (1996), Zhou and Huang (2004) and Hinkley (2003).

Fitting trigonometric models to the climate data, we plot scores of AIC_c , BIC and C_p criteria as functions of the frequency in the left panel of Fig. 1.6. The AIC_c and C_p criteria reach minimum at $\lambda = 2$ and the BIC criterion reaches the minimum at $\lambda = 1$. For a grid of θ in the interval $[0, \log n]$, we calculate the optimal λ , $\hat{\lambda}(\theta)$, based on (1.20). We also calculate the estimated gdf using $T = 1000$ and $\tau = 0.75\bar{\sigma}$. The middle panel of Fig. 1.6 shows the estimated gdf

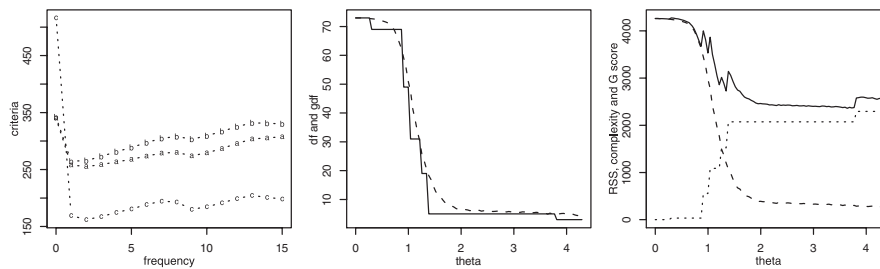


Figure 1.6. Left: Scores of AIC_c , BIC and C_p criteria marked as letters “a”, “b” and “c” respectively. Middle: degrees of freedom (solid line) and estimated gdf (dashed line). Right: RSS (dotted line), model complexity part (dashed line) and the G score (solid line) in (1.22)

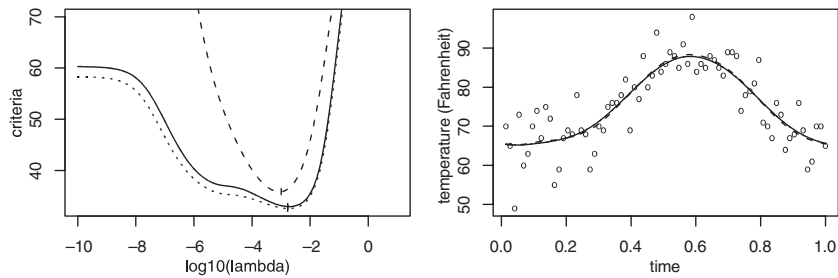


Figure 1.7. Left: Scores of the C_p (UBR), GCV and GML criteria plotted as dotted, solid and dashed lines respectively. Minimum points are marked with vertical bars. Right: Observations (circles) and fits from the periodic spline with the UBR (dotted line), GCV (solid line) and GML (dashed line) choices of smoothing parameter

together with the degrees of freedom based on the selected model, $2\hat{\lambda}(\theta) + 1$. The gdf is intended to account for the extra cost for estimating λ . As expected, the gdf is almost always larger than the degrees of freedom. The gdf is close to the degrees of freedom when θ is small or large. In the middle, it can have significant corrections to the degrees of freedom. Overall, the gdf smoothes out the corners in the discrete degrees of freedom. The RSS, complexity $2\hat{g}_0\hat{\sigma}^2$ and $G(\theta)$ are plotted in the right panel of Fig. 1.6. The minimum of $G(\theta)$ is reached at $\theta = 3.68$ with $\hat{\lambda}(3.68) = 2$. Trigonometric model fits with $\lambda = 1$ and $\lambda = 2$ are shown in Fig. 1.2.

Fitting periodic spline models to the climate data, we plot the C_p (UBR) criterion in the left panel of Fig. 1.7. Fits with the UBR choice of the smoothing parameter is shown in the right panel of Fig. 1.7.

Cross-Validation and Generalized Cross-Validation

The reason one cannot use the GOF for model selection is that it generally underestimates the generalization error of a model (Efron, 1986; Hastie et al., 2002). For example, (1.21) shows that the RSS under-estimates the PSE by $2\sigma^2 \text{tr}H(\lambda)/n$. Thus, similar to the correction term in the AIC, the second term in the C_p criterion corrects this bias. The bias in RSS is a result of using the same data for model fitting and model evaluation. Ideally, these two tasks should be separated using independent samples. This can be achieved by splitting the whole data into two subsamples, a training (calibration) sample for model fitting and a test (validation) sample for model evaluation. This approach, however, is not efficient unless the sample size is large. The idea behind the cross-validation is to recycle data by switching the roles of training and test samples.

Suppose that one has decided on a measure of discrepancy for model evaluation, for example the prediction error. A V -fold cross-validation selects a model as follows.

1. Split the whole data into V disjoint subsamples S_1, \dots, S_V .
2. For $\nu = 1, \dots, V$, fit model M_λ to the training sample $\cup_{i \neq \nu} S_i$, and compute discrepancy, $d_\nu(\lambda)$, using the test sample S_ν .
3. Find optimal λ as the minimizer of the overall discrepancy $d(\lambda) = \sum_{\nu=1}^V d_\nu(\lambda)$.

The cross-validation is a general procedure that can be applied to estimate tuning parameters in a wide variety of problems. To be specific, we now consider the regression model (1.2). For notational simplicity, we consider the delete-1 (leave-one-out) cross-validation with $V = n$. Suppose our objective is prediction. Let \mathbf{y}^{-i} be the $n - 1$ vector with the i th observation, y_i , removed from the original response vector \mathbf{y} . Let \widehat{f}_λ^{-i} be the estimate based on $n - 1$ observations \mathbf{y}^{-i} . The ordinary cross-validation (OCV) estimate of the prediction error is

$$\text{OCV}(\lambda) = \frac{1}{n} \sum_{i=1}^n (y_i - \widehat{f}_\lambda^{-i}(t_i))^2. \quad (1.24)$$

A cross-validation estimate of λ is the minimizer of (1.24). The cross-validation method was introduced by Allen (1974) (also called PRESS) in the context of linear regression and by Wahba and Wold (1975) in the context of smoothing splines. To compute the OCV score, one needs to fit model M_λ n times, once for each delete-one data \mathbf{y}^{-i} . This is computationally intensive. Fortunately, a short-cut exists for many situations. Let

$$\tilde{y}_{ij} = \begin{cases} y_j, & j \neq i, \\ \widehat{f}_\lambda^{-i}(t_i), & j = i, \end{cases}$$

and $\tilde{\mathbf{y}}_i = (\tilde{y}_{i1}, \dots, \tilde{y}_{in})^\top$. $\tilde{\mathbf{y}}_i$ simply replaces the i th element in \mathbf{y} , the one deleted to get \mathbf{y}^{-i} , by $\widehat{f}_\lambda^{-i}(t_i)$. Let \tilde{f}_λ^{-i} be the estimate of f with data $\tilde{\mathbf{y}}_i$. For many methods such as the trigonometric regression (linear regression in general) and periodic spline (smoothing spline in general), we have the following

1 Lemma 1: *Leaving-Out-One Lemma* $\tilde{f}_\lambda^{-i}(t_i) = \widehat{f}_\lambda^{-i}(t_i)$, $i = 1, \dots, n$.

See Wahba (1990) and Hastie and Tibshirani (1990) for proofs. Note that even though it is called the leaving-out-one lemma, similar results hold for the leaving-out-of-cluster cross-validation (Wang et al., 2000). See also Xiang and Wahba (1996), Zhang et al. (2002) and Ke and Wang (2002) for the leaving-out-one lemma for more complicated problems.

For trigonometric regression and periodic spline models, $\widehat{f}_\lambda = H(\lambda)y$ for any y . Thus when y is replaced by \tilde{y}_i , we have $(\tilde{f}_\lambda^{-i}(t_1), \dots, \tilde{f}_\lambda^{-i}(t_n))^T = H(\lambda)\tilde{y}_i$. Denote the elements of $H(\lambda)$ as h_{ij} , $i, j = 1, \dots, n$. Then

$$\widehat{f}_\lambda(t_i) = \sum_{j=1}^n h_{ij}y_j,$$

$$\widehat{f}_\lambda^{-i}(t_i) = \tilde{f}_\lambda^{-i}(t_i) = \sum_{j=1}^n h_{ij}\tilde{y}_j = \sum_{j \neq i} h_{ij}y_j + h_{ii}\widehat{f}_\lambda^{-i}(t_i).$$

Combined, we have

$$\widehat{f}_\lambda(t_i) - \widehat{f}_\lambda^{-i}(t_i) = h_{ii}(y_i - \widehat{f}_\lambda^{-i}(t_i)).$$

Then it is easy to check that

$$y_i - \widehat{f}_\lambda^{-i}(t_i) = (y_i - \widehat{f}_\lambda(t_i))/(1 - h_{ii}),$$

and the OCV reduces to

$$\text{OCV}(\lambda) = \frac{1}{n} \sum_{i=1}^n \left(\frac{y_i - \widehat{f}_\lambda(t_i)}{1 - h_{ii}} \right)^2. \tag{1.25}$$

Thus one only needs to fit the model once with the full data and compute the diagonal elements of the $H(\lambda)$ matrix.

Replacing h_{ii} in (1.25) by the average of all diagonal elements, Craven and Wahba (1979) proposed the following *generalized cross-validation* (GCV) criterion

$$\text{GCV}(\lambda) = \frac{\frac{1}{n} \sum_{i=1}^n (y_i - \widehat{f}_\lambda(t_i))^2}{(1 - \text{tr}H(\lambda)/n)^2}. \tag{1.26}$$

It is easy to see that the GCV criterion is a weighted version of OCV with weights $(1 - h_{ii})^2 / (1 - \text{tr}H(\lambda)/n)^2$. When $\text{tr}H(\lambda)/n$ is small, using the approximation $(1 - x)^2 \approx 1 + 2x$,

$$\text{GCV}(\lambda) \approx \frac{1}{n} \sum_{i=1}^n (y_i - \widehat{f}_\lambda(t_i))^2 + \frac{2}{n} \text{tr}H(\lambda) \left[\frac{1}{n} \sum_{i=1}^n (y_i - \widehat{f}_\lambda(t_i))^2 \right].$$

Regarding $\frac{1}{n} \sum_{i=1}^n (y_i - \widehat{f}_\lambda(t_i))^2$ in the second part as an estimate of σ^2 , the GCV is approximately the same as the C_p (UBR) criterion. Originally proposed to reduce the computational burden, the GCV criterion has been found to possess several favorable properties (Golub et al., 1979; Li, 1985, 1986; 1987, Wahba, 1990; Gu, 2002). Sometimes it is difficult to compute each diagonal element in $H(\lambda)$ directly. Nevertheless, it is relatively easy to approximate $\text{tr}H(\lambda)$ using the randomized trace method (Zhang et al., 2002). Thus the GCV criterion may be adopted to many complicated problems (Xiang and Wahba, 1996; Zhang et al., 2002). The

GCV criterion has non-zero probability to select $\lambda = 0$ (interpolation) which may cause problems when the sample size is small. Fortunately, this probability tends to zero exponentially fast as sample size increases (Wahba and Wang, 1993).

For the trigonometric regression,

$$h_{ii} = \frac{1}{n} \left(1 + \sum_{\nu=1}^{\lambda} 2(\sin^2 2\pi\nu t_i + \cos^2 2\pi\nu t_i) \right) = \frac{1}{n}(1 + 2\lambda) = \frac{\text{tr}H(\lambda)}{n}.$$

For the periodic spline,

$$h_{ii} = \frac{1}{n} + \frac{1}{n} \sum_{\nu=1}^K 2(\sin^2 2\pi\nu t_i + \cos^2 2\pi\nu t_i)/(1 + \lambda(2\pi\nu)^4) = \frac{\text{tr}D}{n} = \frac{\text{tr}H(\lambda)}{n}.$$

Thus the OCV and GCV are the same for both cases.

Instead of deleting one observation at a time, one may delete d observations at a time as described in the V-fold cross-validation. We will call such a method as delete- d CV. Shao (1997) classified various model selection criteria into the following three classes:

Class 1: AIC, C_p , delete-1 CV and GCV.

Class 2: Criterion (1.20) with $\theta \rightarrow \infty$ as $n \rightarrow \infty$, and delete- d CV with $d/n \rightarrow 1$.

Class 3: Criterion (1.20) with a fixed $\theta > 2$, and delete- d CV with $d/n \rightarrow \tau \in (0, 1)$.

BIC is a special case of the Class 2. Shao (1997) showed that the criteria in Class 1 are asymptotically valid if there is no fixed-dimensional correct model and the criteria in Class 2 are asymptotically valid when the opposite is true. Methods in Class 3 are compromises of those in Classes 1 and 2. Roughly speaking, criteria in the first class would perform better if the true model is “complex” and the criteria in the second class would do better if the true model is “simple”. See also Zhang (1993) and Shao (1993).

The climate data subset was selected by first dividing 365 days in the year 1990 into 73 five-day periods, and then selecting measurements on the third day in each period as observations. This is our training sample. Measurements excluding these selected 73 days may be used as the test sample. This test sample consists $365 - 73 = 292$ observations. For the trigonometric model with fixed frequency λ , we calculate the prediction error using the test sample

$$\text{PE}(\lambda) = \frac{1}{292} \sum_{i=1}^{292} (y_i - \hat{f}_\lambda(s_i))^2, \quad (1.27)$$

where s_i are time points for observations in the test sample. The prediction errors are plotted in the left panel of Fig. 1.8 where the minimum is reached at $\lambda = 1$. The GCV scores for the trigonometric model is also plotted in the left panel of Fig. 1.8 where the minimum is reached at $\lambda = 2$. The GCV score for the periodic spline and the corresponding fits are plotted in the left and right panels of Fig. 1.7 respectively. As expected, the GCV scores are similar to the UBR scores.

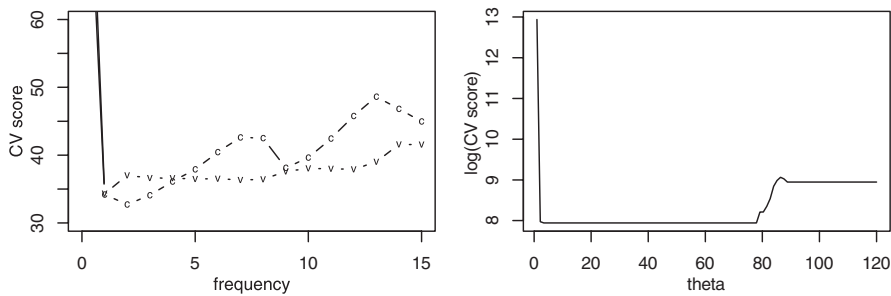


Figure 1.8. Left: prediction errors (1.27) (marked as “v”) and GCV scores (marked as “c”) for the trigonometric model. Right: the OCV scores in (1.28) on the logarithm scale

As a general methodology, the cross-validation may also be used to select θ in (1.20) (Rao and Tibshirani, 1997). Let $\widehat{f}_{\widehat{\lambda}^{-i}(\theta)}^{-i}$ be the estimate based on the delete-one data \mathbf{y}^{-i} where $\widehat{\lambda}^{-i}(\theta)$ is selected using (1.20), also based on \mathbf{y}^{-i} . Then the OCV estimate of prediction error is

$$OCV(\theta) = \frac{1}{n} \sum_{i=1}^n \left(y_i - \widehat{f}_{\widehat{\lambda}^{-i}(\theta)}^{-i}(t_i) \right)^2 . \tag{1.28}$$

The minimum of (1.28) provides an estimate of θ . The OCV score for the trigonometric model is plotted as a function of θ in the right panel of Fig. 1.8. The minimum is reached at a wide range of θ values with $\lambda = 1$ or $\lambda = 2$.

Bayes Factor

Let $P(M_\lambda)$ be the prior probability for model M_λ . For any two models M_{λ_1} and M_{λ_2} , the Bayes factor

$$B(\lambda_1, \lambda_2) = \frac{P(M_{\lambda_1} | \mathbf{y})}{P(M_{\lambda_2} | \mathbf{y})} \div \frac{P(M_{\lambda_1})}{P(M_{\lambda_2})} \tag{1.29}$$

is the posterior odds in favor of model M_{λ_1} divided by the prior odds in favor of model M_{λ_1} (Kass and Raftery, 1995). The Bayes factor provides a scale of evidence in favor of one model versus another. For example, $B(\lambda_1, \lambda_2) = 2$ indicates that the data favor model M_{λ_1} over model M_{λ_2} at odds of two to one. Table 1.1 lists a possible interpretation for Bayes factor suggested by Jeffreys (1961).

The Bayes factor is easy to understand and applicable to a wide range of problems. Methods based on the Bayes factor behave like an Occam’s razor (Jeffreys and Berger, 1992). Non-Bayesian analysis typically selects a model and then proceeds as if the data is generated by the chosen model. Ignoring the fact that the model has been selected from the same data, this approach often leads to under-estimation of the uncertainty in quantities of interest, a problem know as the *model selection*

Table 1.1. Jeffreys' scale of evidence for Bayes factors

Bayes factor	Interpretation
$B(\lambda_1, \lambda_2) < 1/10$	Strong evidence for M_{λ_2}
$1/10 < B(\lambda_1, \lambda_2) < 1/3$	Moderate evidence for M_{λ_2}
$1/3 < B(\lambda_1, \lambda_2) < 1$	Weak evidence for M_{λ_2}
$1 < B(\lambda_1, \lambda_2) < 3$	Weak evidence for M_{λ_1}
$3 < B(\lambda_1, \lambda_2) < 10$	Moderate evidence for M_{λ_1}
$B(\lambda_1, \lambda_2) > 10$	Strong evidence for M_{λ_1}

bias (Chatfield, 1995). Specifically, the estimates of parameters based on the selected model are biased and their variances are usually too optimistic. The Bayesian approach accounts for model uncertainty with the posterior probability $P(M_\lambda | \mathbf{y})$. For example, to predict a new observation y^+ , the best prediction under squared loss is

$$E(y^+ | \mathbf{y}) = \sum_{\lambda \in \Lambda} E(y^+ | M_\lambda, \mathbf{y}) P(M_\lambda | \mathbf{y}),$$

a weighted average of predictions from all models with weights equal to the posterior probabilities. Instead of using a single model, such model averaging incorporates model uncertainty. It also indicates that selecting a single model may not be desirable or necessary for some applications such as prediction (Hoeting et al., 1999).

The practical implementation of Bayesian model selection is, however, far from straightforward. In order to compute the Bayes factor (1.29), one needs to specify priors $P(M_\lambda)$ as well as priors for parameters in each model. While providing a way to incorporating other information into the model and model selection, these priors may be hard to set in practice, and standard non-informative priors for parameters cannot be used (Berger and Pericchi, 1996; Gelman et al., 1995). See Kass and Raftery (1995), Chipman et al. (2001) and Berger and Pericchi (2001) for more discussions on the choice of priors.

After deciding on priors, one needs to compute (1.29) which can be re-expressed as

$$B(\lambda_1, \lambda_2) = \frac{P(\mathbf{y} | M_{\lambda_1})}{P(\mathbf{y} | M_{\lambda_2})}, \quad (1.30)$$

where $P(\mathbf{y} | M_\lambda)$ is the marginal likelihood. The marginal likelihood usually involves an integral which can be evaluated analytically only for some special cases. When the marginal likelihood does not have a closed form, several methods for approximation are available including Laplace approximation, importance sampling, Gaussian quadrature and Markov chain Monte Carlo (MCMC) simulations. Details about these methods are out of the scope of this chapter. References can be found in Kass and Raftery (1995).

Under certain conditions, Kass and Wasserman (1995) showed that

$$-2 \log P(y|M_\lambda) \approx -2 \log(\text{maximum likelihood}) + |M_\lambda| \log n.$$

Thus the BIC is an approximation to the Bayes factor.

In the following we discuss selection of the smoothing parameter λ for the periodic spline. Based on (1.30), our goal is to find λ which maximizes the marginal likelihood $P(y|M_\lambda)$, or equivalently, $P(\tilde{y}|M_\lambda)$ where \tilde{y} is the discrete Fourier transformation of y . Note that

$$\tilde{y} = \alpha + \tilde{\epsilon}, \tag{1.31}$$

where $\tilde{\epsilon} = X_K^\top \epsilon/n \sim N(0, \sigma^2 I/n)$. Let $b = \sigma^2/n\lambda$. Assume the following prior for α :

$$\begin{aligned} \alpha_1 &\propto 1, \\ \alpha_{2\nu} &\sim N(0, b(2\pi\nu)^{-4}), \quad \nu = 1, \dots, K, \\ \alpha_{2\nu+1} &\sim N(0, b(2\pi\nu)^{-4}), \quad \nu = 1, \dots, K, \end{aligned} \tag{1.32}$$

where α_i are mutually independent and are independent of $\tilde{\epsilon}$. An improper prior is assumed for α_1 . It is not difficult to check that $E(\alpha|\tilde{y}) = \hat{\alpha}$. Thus the posterior means of the Bayes model (1.31) and (1.32) are the same as the periodic spline estimates.

Let $z = (\tilde{y}_2, \dots, \tilde{y}_n)^\top$ and write $P(\tilde{y}|M_\lambda) = P(\tilde{y}_1|M_\lambda)P(z|M_\lambda)$. Since $P(\tilde{y}_1|M_\lambda)$ is independent of λ , we will estimate λ using the marginal likelihood $P(z|M_\lambda)$. Since $\tilde{y}_{2\nu}$ or $\tilde{y}_{2\nu+1} \sim N(0, b((2\pi\nu)^{-4} + \lambda))$, the log marginal likelihood of z is

$$l(b, \lambda) = -\frac{n-1}{2} \log 2\pi - \frac{n-1}{2} \log b - \sum_{\nu=1}^K \log[(2\pi\nu)^{-4} + \lambda] - \frac{1}{2b} \sum_{\nu=1}^K \frac{\tilde{y}_{2\nu}^2 + \tilde{y}_{2\nu+1}^2}{(2\pi\nu)^{-4} + \lambda}.$$

Fixing λ and maximizing with respect to b , we have

$$\hat{b} = \frac{1}{n-1} \sum_{\nu=1}^K \frac{\tilde{y}_{2\nu}^2 + \tilde{y}_{2\nu+1}^2}{(2\pi\nu)^{-4} + \lambda}.$$

Plugging back, we have

$$l(\hat{b}, \lambda) = \text{constant} - \frac{n-1}{2} \log \sum_{\nu=1}^K \frac{\tilde{y}_{2\nu}^2 + \tilde{y}_{2\nu+1}^2}{(2\pi\nu)^{-4} + \lambda} - \sum_{\nu=1}^K \log [(2\pi\nu)^{-4} + \lambda].$$

Thus maximizing the log likelihood is equivalent to minimizing

$$M(\lambda) = \frac{\sum_{\nu=1}^K (\tilde{y}_{2\nu}^2 + \tilde{y}_{2\nu+1}^2) / ((2\pi\nu)^{-4} + \lambda)}{\left(\prod_{\nu=1}^K ((2\pi\nu)^{-4} + \lambda)\right)^{2/(n-1)},$$

It is not difficult to check that

$$M(\lambda) = \frac{\mathbf{y}^\top (I - A(\lambda)) \mathbf{y}}{(\det^+(I - A(\lambda)))^{1/(n-1)}}, \quad (1.33)$$

where \det^+ is the product of non-zero eigenvalues. The criterion (1.33) is called the *generalized maximum likelihood* method in smoothing spline literature (Wahba, 1990). It is the same as the *restricted maximum likelihood* (REML) method in the mixed effects literature (Wang, 1998). Note that the marginal likelihood is approximated by plugging-in \hat{b} rather than averaging over a prior distribution for b .

For the climate data, the GML scores for the periodic spline and the corresponding fits are plotted in the left and right panels of Fig. 1.7 respectively. The fits with three different choices of the smoothing parameter are very similar.

Impact of Heteroscedasticity and Correlation

1.6

In our climate example we used one fifth of all measurements in the year 1990. Figure 1.9 shows all measurements in 1990 and periodic spline fits using all measurements with GCV, GML and UBR choices of the smoothing parameter. Obviously the GCV and UBR criteria under-estimate the smoothing parameter which leads to wiggly fits. What is causing the GCV and UBR methods to breakdown?

In model (1.2) we have assumed that random errors are iid with mean zero and variance σ^2 . The middle panel of Fig. 1.1 indicates that variation of the maximum temperature is larger during the winter. Also, temperatures close in time may be correlated. Thus the assumption of homoscedasticity and independence may not hold. What kind of impact, if any, do these potential violations have on the model selection procedures?

For illustration, we again consider two simulations with heteroscedastic and auto-correlated random errors respectively. We use the same function and de-

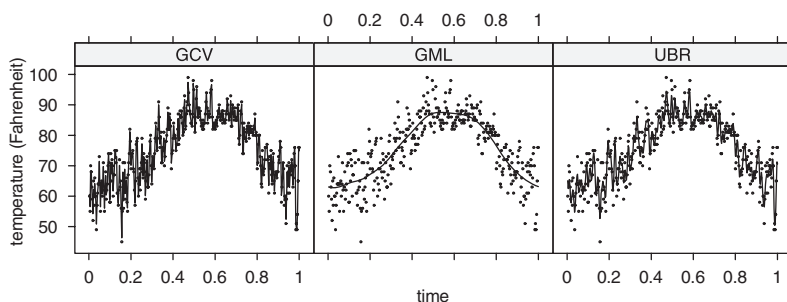


Figure 1.9. Points are all measurements in 1990. Lines are the periodic spline estimates. The methods for selecting the smoothing parameter are indicated in strips

sign points as the simulation in Sect. 1.2 with the true function shown in the left panel of Fig. 1.4. For heteroscedasticity, we generate random errors $\varepsilon_i \sim N(0, ((i + 36.5)/147)^2)$, $i = 1, \dots, 73$, where the variance increases with i . For correlation, we generate the ε_i 's as a first-order autoregressive process with mean zero, standard deviation 0.5 and first-order correlation 0.5. The first and the second rows in Fig. 1.10 show the fits by the trigonometric model with cross-validation, BIC and C_p choices of orders under heteroscedastic and auto-correlated random errors respectively but without adjustment for the heteroscedasticity or correlation. The third and the fourth rows in Fig. 1.10 show the fits by the periodic spline with

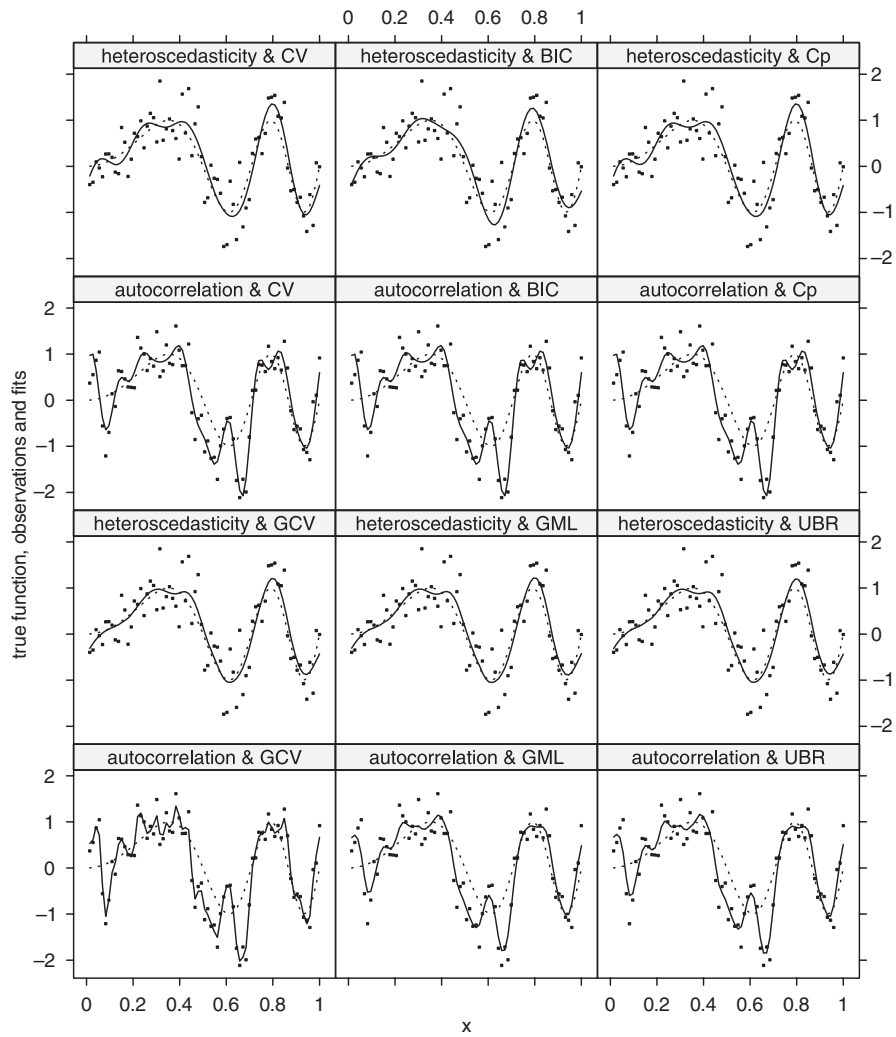


Figure 1.10. Circles are observations. Dotted lines are true functions. Solid lines are the estimated functions. Simulation settings and model selection criteria are marked in strips

GCV, GML and UBR choices of smoothing parameters under heteroscedastic and auto-correlated random errors respectively but without adjustment for the heteroscedasticity or correlation. These kind of fits are typical under two simulation settings. The heteroscedasticity has some effects on the model selection, but far less severe than the impact of auto-correlation. It is well-known that positive auto-correlation leads to under-smoothing for non-parametric models with data-driven choices of the smoothing parameter (Wang, 1998; Opsomer et al., 2001). Figure 1.10 shows that the same problem exists for parametric regression models as well.

The breakdown of the GCV and UBR criteria for the climate data is likely caused by the auto-correlation which is higher when daily measurements are used as observations. Extensions of the GCV, GML and UBR criteria for correlated data can be found in Wang (1998).

1.7

Discussion

There are many fundamental and philosophical issues in model selection. For example, a model is usually a simplification or approximation of the complicated reality. “All models are wrong, but some are useful” (Box, 1976). A selected model tell us what the finite data are likely to support, not the full reality.

Data analysts are constantly making model selections (assumptions), consciously or unconsciously. For example, certain choices have to be made for selecting the candidate models \mathcal{M} (Burnham and Anderson, 2002). The selection methods have been formalized in the current literature represent only a fraction of the whole selection process in practice. As a consequence, model selection is considered as both science and art. Scientific knowledge, empirical evidence, common sense, and good judgment all play an important role in this process. It is rarely the case that sufficient information is available to fully specify the model. Thus creative, critical and careful thinking is required. The problem is often so complicated that one should not expect to achieve the final model in one attempt, regardless of which model selection method has been used. Instead, an iterative scheme including diagnostics suggested by Box and Jenkins (1976) should be used.

Some methods such as cross-validation can be applied to a wide variety of applications, while others are designed for specific applications. Different methods have been motivated and justified with different target criteria under different assumptions. Thus it is unrealistic to expect one method to serve all purposes and perform uniformly better under all circumstances.

We used prediction criteria including MSE and PSE as examples. This, of course, does not mean model selection is involved in (or for) prediction only. For example, another important objective of a model is data description. Identifying risk factors for diabetes is as important as predicting a person’s chance of having this disease.

Wang and Ke (2002) have developed a user-friendly S package, ASSIST, which includes several functions for fitting various spline based models. The `ssr` function in this package is used to fit periodic spline models in this chapter. The ASSIST pack-

age can be downloaded from <http://www.pstat.ucsb.edu/faculty/yuedong/software>. More details and examples can be found in the manual of the ASSIST package which also is available at this web-site.

Acknowledgments. This work was supported by NIH Grants R01 GM58533.

References

- Akaike, H. (1970). Statistical predictor identification. *Ann. Inst. Statist. Math.*, 21: 203–217.
- Akaike, H. (1973). Information theory and the maximum likelihood principle. *International Symposium on Information Theory*, eds. V. Petrov and F. Csáki, Budapest: Akademiai Kiádo, pp. 267–281.
- Allen, D. M. (1974). The relationship between variable selection and data augmentation and a method for prediction. *Technometrics*, 16: 125–127.
- Bai, Z. D., Rao, C. R. and Wu, Y. (1999). Model selection with data-oriented penalty. *Journal of Statistical Planning and Inference*, 77: 103–117.
- Beran, R. (1996). Bootstrap variable selection and confidence sets. In Rieder, H. (ed), *Robust Statistics, Data Analysis and Computer Intensive Methods*, Springer Lecture Notes in Statistics 109.
- Berger, J. O. and Pericchi, L. R. (1996). The intrinsic bayes factor for model selection and prediction. *Journal of the American Statistical Association*, 91: 109–122.
- Berger, J. O. and Pericchi, L. R. (2001). Objective bayesian methods for model selection: Introduction and comparison. In Lahiri, P. (ed), *Model Selection*, Institute of Mathematical Statistics Lecture Notes – Monograph Series volume 38, pp.135–207, Beachwood Ohio.
- Box, G. E. P. (1976). Science and statistics. *Journal of the American Statistical Association*, 71: 791–799.
- Box, G. E. P. and Jenkins, G. M. (1976). *Time Series Analysis*, Holden-Day.
- Burnham, K. P. and Anderson, D. R. (2002). *Model Selection and Multimodel Inference*, 2nd ed., Springer, New York.
- Chatfield, C. (1995). Model uncertainty, data mining and statistical inference (with discussion). *Journal of the Royal Statistical Society B*, 158: 419–466.
- Chipman, H., George, E. I. and McCulloch, R. E. (2001). The practical implementation of bayesian model selection. In Lahiri, P. (ed), *Model Selection*, Institute of Mathematical Statistics Lecture Notes – Monograph Series volume 38, pp.65–134, Beachwood Ohio.
- Craven, P. and Wahba, G. (1979). Smoothing noisy data with spline functions. *Numer. Math.*, 31: 377–403.
- Dette, H., Munk, A. and Wagner, T. (1998). Estimating the variance in nonparametric regression – what is a reasonable choice? *Journal of the Royal Statistical Society B*, 60: 751–764.

- Donoho, D. L. and Johnston, I. M. (1994). Ideal spatial adaption by wavelet shrinkage. *Biometrika*, 81: 425–456.
- Efron, B. (1986). How biased is the apparent error rate of a prediction rule. *Journal of the American Statistical Association*, 81: 461–470.
- Gasser, T., Sroka, L. and Jennen-Steinmetz, C. (1986). Residual variance and residual pattern in nonlinear regression. *Biometrika*, 73: 625–633.
- Gelman, A., Carlin, J. B., Stern, H. S. and Rubin, D. B. (1995). *Bayesian Data Analysis*, Chapman & Hall, Boca Raton.
- George, E. I. (2000). The variable selection problem. *Journal of the American Statistical Association*, 95: 1304–1308.
- Golub, G., Heath, M. and Wahba, G. (1979). Generalized cross validation as a method for choosing a good ridge parameter. *Technometrics*, 21: 215–224.
- Gu, C. (1998). Model indexing and smoothing parameter selection in non-parametric function estimation (with discussion). *Statistica Sinica*, 8: 632–638.
- Gu, C. (2002). *Smoothing Spline ANOVA Models*, Springer-Verlag, New York.
- Hall, P., Kay, J. W. and Titterton, D. M. (1990). Asymptotically optimal difference-based estimation of variance in nonparametric regression. *Biometrika*, 77: 521–528.
- Hastie, T. and Tibshirani, R. (1990). *Generalized Additive Models*, Chapman and Hall.
- Hastie, T., Tibshirani, R. and Friedman, J. (2002). *The Elements of Statistical Learning*, Springer, New York.
- Hinkley, D. (2003). Bootstrap methods for variable selection and shrinkage estimation confidence sets, Personal Communication.
- Hoeting, J. A., Madigan, D., Raftery, A. E. and Volinsky, C. T. (1999). Bayesian model averaging: a tutorial (with discussion). *Statistical Science*, 14: 382–417. Corrected version available at <http://www.stat.washington.edu/www/research/online/hoeting1999.pdf>.
- Hurvich, C. M. and Tsai, C. L. (1989). Regression and time series model selection in small samples. *Biometrika*, 76: 297–207.
- Jeffreys, H. (1961). *Theory of Probability*, Oxford: Clarendon Press.
- Jeffreys, W. and Berger, J. O. (1992). Ockham's razor and bayesian analysis. *American Scientist*, 80: 64–72.
- Kass, R. E. and Raftery, A. (1995). Bayesian factors. *Journal of the American Statistical Association*, 90: 773–795.
- Kass, R. E. and Wasserman, L. (1995). A reference bayesian test for nested hypotheses and its relationship to the schwarz criterion. *Journal of the American Statistical Association*, 90: 982–934.
- Ke, C. and Wang, Y. (2002). Nonparametric nonlinear regression models, Technical Report # 385, Department of Statistics and Applied Probability, University of California, Santa Barbara.
- Li, K. C. (1985). From Stein's unbiased risk estimates to the method of generalized cross-validation. *Annals of Statistics*, 13: 1352–1377.

- Li, K. C. (1986). Asymptotic optimality of C_L and generalized cross-validation in ridge regression with application to spline smoothing. *Annals of Statistics*, 14: 1101–1112.
- Li, K. C. (1987). Asymptotic optimality of C_p , C_L , cross-validation and generalized cross-validation: Discrete index set. *Annals of Statistics*, 15: 958–975.
- Linhart, H. and Zucchini, W. (1986). *Model Selection*, Wiley, New York.
- Mallows, C. L. (1973). Some comments on C_p . *Technometrics*, 12: 661–675.
- Miller, A. (2002). *Subset Selection in Regression, 2nd ed.*, Chapman & Hall, New York.
- Opsomer, J., Wang, Y. and Yang, Y. (2001). Nonparametric regression with correlated errors. *Statistical Science*, 16: 134–153.
- Rao, C. R. and Wu, Y. (1989). A strongly consistent procedure for model selection in a regression problem. *Biometrika*, 76: 369–374.
- Rao, J. S. (1999). Bootstrap choice of cost complexity for better subset selection. *Statistica Sinica*, 9: 273–287.
- Rao, J. S. and Tibshirani, R. (1997). Discussion to “an asymptotic theory for model selection” by Jun Shao. *Statistica Sinica*, 7: 249–252.
- Rice, J. A. (1984). Bandwidth choice for nonparametric regression. *Annals of Statistics*, 12: 1215–1230.
- Schwarz, G. (1978). Estimating the dimension of a model. *Annals of Statistics*, 12: 1215–1231.
- Shao, J. (1993). Linear model selection by cross-validation. *Journal of the American Statistical Association*, 88: 486–494.
- Shao, J. (1997). An asymptotic theory for linear model selection (with discussion), *Statistica Sinica*, 7: 221–264.
- Shen, X. and Ye, J. (2002). Adaptive model selection. *Journal of the American Statistical Association*, 97: 210–221.
- Stone, M. (1974). Cross-validated choice and assessment of statistical prediction, *Journal of the Royal Statistical Society B*, 36: 111–147.
- Wahba, G. (1990). *Spline Models for Observational Data*, SIAM, Philadelphia. CBMS-NSF Regional Conference Series in Applied Mathematics, Vol. 59.
- Wahba, G. and Wang, Y. (1993). Behavior near zero of the distribution of GCV smoothing parameter estimates for splines. *Statistics and Probability Letters*, 25: 105–111.
- Wahba, G. and Wold, S. (1975). A completely automatic french curve. *Communications in Statistics*, 4: 1–17.
- Wang, Y. (1998). Smoothing spline models with correlated random errors. *Journal of the American Statistical Association*, 93: 341–348.
- Wang, Y. and Ke, C. (2002). ASSIST: A suite of s-plus functions implementing spline smoothing techniques, Manual for the ASSIST package. Available at <http://www.pstat.ucsb.edu/faculty/yuedong/software>.
- Wang, Y., Guo, W. and Brown, M. B. (2000). Spline smoothing for bivariate data with applications to association between hormones. *Statistica Sinica*, 10: 377–397.
- Xiang, D. and Wahba, G. (1996). A generalized approximate cross validation for smoothing splines with non-gaussian data. *Statistica Sinica*, 6: 675–692.

- Yang, Y. (1999). Model selection for nonparametric regression. *Statistica Sinica*, 9: 475–499.
- Ye, J. (1998). On measuring and correcting the effects of data mining and model selection. *Journal of the American Statistical Association*, 93: 120–131.
- Zhang, H., Wahba, G., Lin, Y., Voelker, M., Ferris, M., Klein, R. and Klein, B. (2002). Variable selection and model building via likelihood basis pursuit, Technical Report No. 1059, Department of Statistics, University of Wisconsin.
- Zhang, P. (1993). Model selection via multifold cross validation. *Annals of Statistics*, 21: 299–313.
- Zhou, H. and Huang, J. T. (2004). Shrinkage estimation toward the data chosen reduced model with applications to wavelet analysis and principal component regression analysis. *Annals of Statistics*, to appear.

Bootstrap and Resampling

III.2

Enno Mammen, Swagata Nandi

2.1	<i>Introduction</i>	469
2.2	<i>Bootstrap as a Data Analytical Tool</i>	473
2.3	<i>Resampling Tests and Confidence Intervals</i>	479
2.4	<i>Bootstrap for Dependent Data</i>	484
	The Subsampling	484
	The Block Bootstrap	485
	The Sieve Bootstrap	486
	The Nonparametric Autoregressive Bootstrap	487
	The Regression-type Bootstrap, the Wild Bootstrap and the Local Bootstrap	487
	The Markov Bootstrap	489
	The Frequency Domain Bootstrap	490

Introduction

The bootstrap is by now a standard method in modern statistics. Its roots go back to a lot of resampling ideas that were around in the seventies. The seminal work of Efron (1979) synthesized some of the earlier resampling ideas and established a new framework for simulation based statistical analysis. The idea of the bootstrap is to develop a setup to generate more (pseudo) data using the information of the original data. True underlying sample properties are reproduced as closely as possible and unknown model characteristics are replaced by sample estimates.

In its basic nature the bootstrap is a data analytic tool. It allows to study the performance of statistical methods by applying them repeatedly to bootstrap pseudo data (“resamples”). The inspection of the outcomes for the different bootstrap resamples allows the statistician to get a good feeling on the performance of the statistical procedure. In particular, this concerns graphical methods. The random nature of a statistical plot is very difficult to be summarized by quantitative approaches. In this respect data analytic methods differ from classical estimation and testing problems. We will illustrate data analytic uses of the bootstrap in the next section.

Most of the bootstrap literature is concerned with bootstrap implementations of tests and confidence intervals and bootstrap applications for estimation problems. It has been argued that for these problems bootstrap can be better understood if it is described as a plug-in method. Plug-in method is an approach used for the estimation of functionals that depend on unknown finite or infinite dimensional model parameters of the observed data set. The basic idea of plug-in estimates is to estimate these unknown parameters and to plug them into the functional. A wide known example is the plug-in bandwidth selector for kernel estimates. Asymptotical optimal bandwidths typically depend e.g. on averages of derivatives of unknown curves (e.g. densities, regression functions), residual variances, etc. Plug-in bandwidth selectors are constructed by replacing these unknown quantities by finite sample estimates. We now illustrate why the bootstrap can be understood as a plug-in approach. We will do this for i.i.d. resampling. This is perhaps the most simple version of the bootstrap. It is applied to an i.i.d. sample X_1, \dots, X_n with underlying distribution P . I.i.d. resamples are generated by drawing n times with replacement from the original sample X_1, \dots, X_n . This gives a resample X_1^*, \dots, X_n^* . More formally, the resample is constructed by generating X_1^*, \dots, X_n^* that are conditionally independent (given the original data set) and have conditional distribution \widehat{P}_n . Here \widehat{P}_n denotes the empirical distribution. This is the distribution that puts mass $1/n$ on each value of X_1, \dots, X_n in case that all observations have different values (or more generally, mass j/n on points that appear j times in the sample), i.e. for a set A we have $\widehat{P}_n(A) = n^{-1} \sum_{i=1}^n I(X_i \in A)$ where I denotes the indicator function. The bootstrap estimate of a functional $T(P)$ is defined as the plug-in estimate $T(\widehat{P}_n)$. Let us consider the mean $\mu(P) = \int xP(dx)$ as a simple example. The bootstrap estimate of $\mu(P)$ is given by $\mu(\widehat{P}_n)$. Clearly, the bootstrap estimate is equal to the sample mean $\bar{X}_n = n^{-1} \sum_{i=1}^n X_i$. In this simple case, simulations are not needed to

calculate the bootstrap estimate. Also in more complicated cases it is very helpful to distinguish between the statistical performance and the algorithmic calculation of the bootstrap estimate. In some cases it may be more appropriate to calculate the bootstrap estimate by Monte-Carlo simulations, in other cases powerful analytic approaches may be available. The discussion which algorithmic approach is preferable should not be mixed up with the discussion of the statistical properties of the bootstrap estimate. Perhaps, clarification of this point is one of the major advantages of viewing the bootstrap as a plug-in method. Let us consider now a slightly more complicated example. Suppose that the distribution of $\sqrt{n}[\bar{X}_n - \mu(P)]$ is our functional $T_n(P) = T(P)$ that we want to estimate. The functional now depends on the sample size n . The factor \sqrt{n} has been introduced to simplify asymptotic considerations following below. The bootstrap estimate of $T_n(P)$ is equal to $T_n(\hat{P}_n)$. This is the conditional distribution of $\sqrt{n}[\bar{X}_n^* - \mu(\hat{P}_n)] = \sqrt{n}(\bar{X}_n^* - \bar{X}_n)$, given the original sample X_1, \dots, X_n . In this case the bootstrap estimate could be calculated by Monte-Carlo simulations. Resamples are generated repeatedly, say m -times, and for the j -th resample the bootstrap statistic $\Delta_j = \sqrt{n}(\bar{X}_n^* - \bar{X}_n)$ is calculated. This gives m values $\Delta_1, \dots, \Delta_m$. Now the bootstrap estimate $T_n(\hat{P}_n)$ is approximated by the empirical distribution of these m values. E.g. the quantiles of the distribution $T_n(P)$ of $\sqrt{n}[\bar{X}_n - \mu(P)]$ are estimated by the sample quantiles of $\Delta_1, \dots, \Delta_m$. The bootstrap quantiles can be used to construct “bootstrap confidence intervals” for $\mu(P)$. We will come back to bootstrap confidence intervals in Sect. 2.3.

There are two other advantages of the plug-in view of the bootstrap. First, the estimate of P that is plugged into the functional T_n could be replaced by other estimates. For example if one is willing to assume that the observations have a symmetric distribution around their mean one could replace \hat{P}_n by a symmetrized version. Or if one is using a parametric model $\{P_\theta : \theta \in \Theta\}$ for the observations one could use $P_{\hat{\theta}}$ where $\hat{\theta}$ is an estimate of the parameter θ . In the latter case one also calls the procedure parametric bootstrap. In case that the parametric model holds one may expect a better accuracy of the parametric bootstrap whereas, naturally, the “nonparametric” bootstrap is more robust against deviations from the model. We now come to another advantage of the plug-in view. It gives a good intuitive explanation when the “bootstrap works”. One says that the bootstrap works or bootstrap is consistent if the difference between $T_n(\hat{P}_n)$ and $T_n(P)$, measured by some distance, converges to zero. Here \tilde{P}_n is some estimate of P . The Bootstrap will work when two conditions hold:

- (1) The estimate \tilde{P}_n is a consistent estimate of P , i.e. \tilde{P}_n converges to P , in some appropriate sense.
- (2) The functionals T_n are continuous, uniformly in n .

Consistency of the bootstrap has been proved for a broad variety of models and for a large class of different bootstrap resampling schemes. Typically for the proofs another approach has been used than (1) and (2). Using asymptotic theory often one can verify that $T_n(\tilde{P}_n)$ and $T_n(P)$ have the same limiting distribution, see Bickel and Freedman (1981) for one of the first consistency proofs for the bootstrap. In our example if the observations have a finite variance $\sigma^2(P)$ then both $T_n(\tilde{P}_n)$

and $T_n(P)$ have a limiting normal limit $N(0, \sigma^2(P))$. For a more general discussion of the approach based on (1) and (2), see also Beran and Ducharme (1991). The importance of (1) and (2) also lies in the fact that it gives an intuitive reasoning when the bootstrap works. For a recent discussion if assumption (2) is necessary see also Inoue and Kilian (2003).

There exist bootstrap methods that cannot be written or interpreted as plug-in estimates. This concerns different bootstrap methods where random weights are generated instead of random (pseudo) observations (see Bose and Chatterjee, 2002). Or this may happen in many applications where the data model is not fully specified. Important examples are models for dependent data. Whereas classical parametric time series models specify the full dimensional distribution of the complete data vector, some non- and semi-parametric models only describe the distribution of neighbored observations. Then the full data generating process is not specified and a basic problem arises how bootstrap resamples should be generated. There are some interesting proposals around and the research on bootstrap for dependent data is still going on. We give a short introduction to this topic in Sect. 2.4. It is a nice example of an active research field on the bootstrap.

Several reasons have been given why the bootstrap should be applied. The Bootstrap can be compared with other approaches. In our example the classical approach would be to use the normal approximation $N(0, \sigma^2(\hat{P}_n))$. It has been shown that the bootstrap works if and only if the normal approximation works, see Mammen (1992a). This even holds if the observations are not identically distributed. Furthermore, one can show that the rate of convergence of both the bootstrap and the normal approximation is $n^{-1/2}$. This result can be shown by using Edgeworth expansions. We will give a short outline of the argument. The distribution function $F(x) = P(\sqrt{n}[\bar{X}_n - \mu(P)] \leq x)$ can be approximated by

$$\Phi\left(\frac{x}{\sigma(P)}\right) - \frac{1}{6\sqrt{n}} \frac{\mu_3(P)}{\sigma(P)^3} \left[\left(\frac{x}{\sigma(P)}\right)^2 - 1 \right] \phi\left(\frac{x}{\sigma(P)}\right).$$

Here, Φ is the distribution function of a standard normal distribution and ϕ is its density. $\mu_3(P) = E[X_i - \mu(P)]^3$ is the third centered moment of the observations X_i . Under regularity conditions this approximation holds with errors of order $O(n^{-1})$. For the bootstrap estimate of F a similar expansion can be shown where $\sigma(P)$ and $\mu_3(P)$ are replaced by their sample versions $\sigma(\hat{P}_n)$ and $\mu_3(\hat{P}_n) = n^{-1} \sum_{i=1}^n (X_i - \bar{X}_n)^3$

$$\Phi\left(\frac{x}{\sigma(\hat{P}_n)}\right) - \frac{1}{6\sqrt{n}} \frac{\mu_3(\hat{P}_n)}{\sigma(\hat{P}_n)^3} \left[\left(\frac{x}{\sigma(\hat{P}_n)}\right)^2 - 1 \right] \phi\left(\frac{x}{\sigma(\hat{P}_n)}\right).$$

The difference between the bootstrap estimate and F is of order $n^{-1/2}$ because the first order terms $\Phi(x/\sigma(P))$ and $\Phi(x/\sigma(\hat{P}_n))$ differ by a term of order $O_p(n^{-1/2})$ as the same holds for $\sigma(P)$ and $\sigma(\hat{P}_n)$. Thus there seems to be no asymptotic advantage in using the bootstrap compared to the classical normal approximation although the skewness of the distribution is accurately mimicked by the bootstrap. However, if the functional T_n is replaced by the distribution of the studentized

statistic $\sqrt{n}\sigma(\widehat{P}_n)^{-1}(\overline{X}_n - \mu(P))$ then the bootstrap achieves a rate of convergence of order n^{-1} whereas the normal approximation $N(0, 1)$ still only has a rate of accuracy of order $n^{-1/2}$. Again, this can be easily seen by Edgeworth expansions. For the distribution function of the studentized statistic the following expansion holds with accuracy $O(1/n)$.

$$\Phi(x) + \frac{1}{6\sqrt{n}} \frac{\mu_3(P)}{\sigma(P)^3} [2x^2 + 1] \phi(x) .$$

The normal approximation $\Phi(x)$ differs from this expansion by terms of order $O(n^{-1/2})$. For the bootstrap estimate one gets the following expansion with error terms of order $O(1/n)$.

$$\Phi(x) + \frac{1}{6\sqrt{n}} \frac{\mu_3(\widehat{P}_n)}{\sigma(\widehat{P}_n)^3} [2x^2 + 1] \phi(x) .$$

This approximates the distribution function of the studentized statistic with accuracy $O_P(n^{-1})$ because $\mu_3(\widehat{P}_n) - \mu_3(P) = O_P(n^{-1/2})$ and $\sigma(\widehat{P}_n) - \sigma(P) = O_P(n^{-1/2})$. That means in this case the classical normal approximation is outperformed by the bootstrap. This result for studentized statistics has been used as the main asymptotic argument for the bootstrap. It has been verified for a large class of models and resampling methods. For a rigorous and detailed discussion see Hall (1992).

There also exist some other arguments in favor of the bootstrap. For linear models with increasing dimension it has been shown in Bickel and Freedman (1983) and Mammen (1989,1992b,1993) that the bootstrap works under weaker conditions than the normal approximation. These results have been extended to more general sequences of models and resampling schemes, see Bose and Chatterjee (2002) and references cited therein. These results indicate that the bootstrap still may give reasonable results even when the normal approximation does not work. For many applications this type of result may be more important than a comparison of higher order performances. Higher order Edgeworth expansions only work if the simple normal approximation is quite reasonable. But then the normal approximation is already sufficient for most statistical applications because typically not very accurate approximations are required. For example an actual level .06 instead of an assumed level .05 may not lead to a misleading statistical analysis. Thus one may argue that higher order Edgeworth expansions can only be applied when they are not really needed and for these reasons they are not the appropriate methods for judging the performance of the bootstrap. On the other hand no other mathematical technique is available that works for such a large class of problems as the Edgeworth expansions do. Thus there is no general alternative way for checking the accuracy of the bootstrap and for comparing it with normal approximations.

The Bootstrap is a very important tool for statistical models where classical approximations are not available or where they are not given in a simple form. Examples arise e.g. in the construction of tests and confidence bands in nonparametric curve estimation. Here approximations using the central limit theorem lead

to distributions of functionals of Gaussian processes. Often these distributions are not explicitly given and must be calculated by simulations of Gaussian processes. We will give an example in the next section (number of modes of a kernel smoother as a function of the bandwidth). Compared with classical asymptotic methods the bootstrap offers approaches for a much broader class of statistical problems.

By now, the bootstrap is a standard method of statistics. It has been discussed in a series of papers, overview articles and books. The books Efron (1982), Efron and Tibshirani (1993) and Davison and Hinkley (1997) give a very insightful introduction into possible applications of the bootstrap in different fields of statistics. The books Beran and Ducharme (1991) and Mammen (1992b) contain a more technical treatment of consistency of the bootstrap, see also Gine (1997). Higher order performance of the bootstrap is discussed in the book Hall (1992). The book Shao and Tu (1995) gives a rather complete overview on the theoretical results on the bootstrap in the mid-nineties. The book Politis, Romano, and Wolf (1999) gives a complete discussion of the subsampling, a resampling method where the resample size is smaller than the size of the original sample. The book Lahiri (2003b) discusses the bootstrap for dependent data. Some overview articles are contained in *Statistical Science* (2003), Vol. 18, Number 2. Here, Efron (2003) gives a short (re)discussion of bootstrap confidence intervals, Davison, Hinkley and Young (2003) report on recent developments of the bootstrap, in particular in classification, Hall (2003) discusses the roots of the bootstrap, and Boos (2003) and Beran (2003) give a short introduction to the bootstrap, and other articles give an overview over recent developments of bootstrap applications in different fields of statistics. Overview articles over special bootstrap applications have been given for sample surveys (Shao, 1996, 2003 and Lahiri, 2003a), for econometrics (Horowitz, 1997, 2001, 2003a), nonparametric curve estimation (Härdle and Mammen, 1991, Mammen, 2000), estimating functions (Lele, 2003) and time series analysis (Bühlmann, 2002, Härdle, Horowitz and Kreiss, 2003, Politis, 2003).

2.2 **Bootstrap as a Data Analytical Tool**

In a data analysis the statistician wants to get a basic understanding of the stochastic nature of the data. For this purpose he/she applies several data analytic tools and interprets the results. A basic problem of a data analysis is over-interpretation of the results after a battery of statistical methods has been applied. A similar situation occurs in multiple testing but there exist approaches to capture the joint stochastics of several test procedures. The situation becomes more involved in modern graphical data analysis. The outcomes of a data analytic tool are plots and the interpretation of the data analysis relies on the interpretation of these (random) plots. There is no easy way to have an understanding of the joint distribution of the inspected graphs. The situation is already complicated if only one graph is checked. Typically it is not clearly specified for which characteristics the plot is checked. We will illustrate this by a simple example. We will argue that

the bootstrap and other resampling methods offer a simple way to get a basic understanding for the stochastic nature of plots that depend on random data. In the next section we will discuss how this more intuitive approach can be translated into the formal setting of mathematical decision theoretical statistics. Our example is based on the study of a financial time series. Figure 2.1 shows the daily values of the German DAX index from end of 1993 until November 2003. In Fig. 2.2 mean-corrected log returns are shown. Logreturns for a series x_t are defined as $\log x_t - \log x_{t-1}$. Mean-corrected logreturns r_t are defined as this difference minus its sample average. Under the Black-Sholes model the logreturns r_t are i.i.d. It belongs to folklore in finance that this does not hold. We now illustrate how this could be seen by application of resampling methods. Figure 2.3 shows a plot of the same logreturns as in Fig. 2.2 but with changed order. The logreturns are plotted against a random permutation of the days. The clusters appearing in Fig. 2.2 disappear. Figure 2.3 shows that these clusters could not be explained by stochastic fluctuations. The same story is told in Fig. 2.4. Here a bootstrap sample of the logreturns is shown. Logreturns are drawn with replacement from the set of all logreturns (i.i.d. resampling) and they are plotted in the order as they were drawn. Again the clusters disappear and the same happens for typical repetitions of random permutation or bootstrap plots. The clusters in Fig. 2.2 can be interpreted as volatility clusters. The volatility of a logreturn for a day is defined as the conditional variance of the logreturn given the logreturns of all past days. The volatilities of neighbored days are positively correlated. This results in volatility clusters. A popular approach to model the clusters are

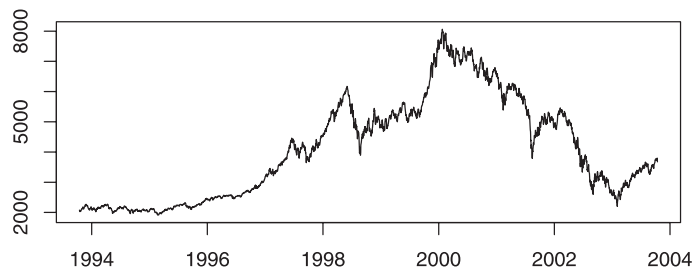


Figure 2.1. Plot of DAX data

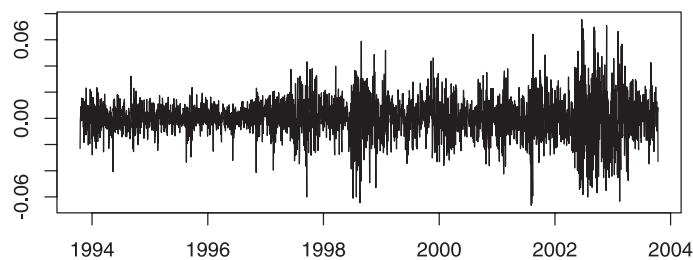


Figure 2.2. 1-lag mean-corrected logreturn of the DAX data

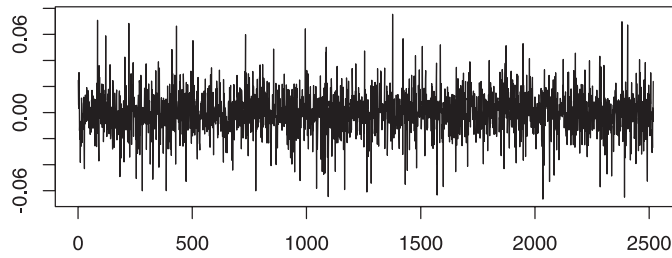


Figure 2.3. Random permutation of the data in Fig. 2.2

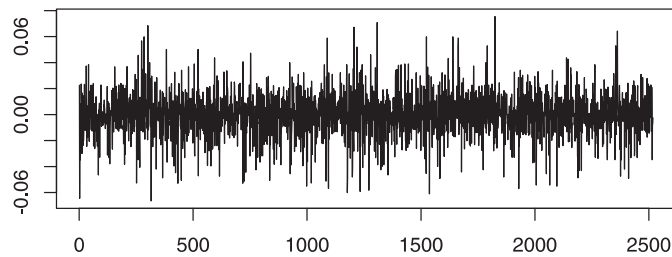


Figure 2.4. Nonparametric bootstrap sample of the data in Fig. 2.2

GARCH (Generalized Autoregressive Conditionally Heteroscedastic) models. In the GARCH(1,1) specification one assumes that $r_t = \sigma_t \varepsilon_t$ where ε_t are i.i.d. errors with mean zero and variance 1 and where σ_t^2 is a random conditional variance process fulfilling $\sigma_t^2 = a_0 + a_1 \sigma_{t-1}^2 + b_1 r_{t-1}^2$. Here a_0 , a_1 and b_1 are unknown parameters. Figure 2.5 shows a bootstrap realization of a fitted GARCH(1,1) model. Fitted parameters \hat{a}_0 , \hat{a}_1 and \hat{b}_1 are calculated by a quasi-likelihood method (i.e. likelihood method for normal ε_t). In the bootstrap resampling the errors ε_t are generated by i.i.d. resampling from the residuals $r_t/\hat{\sigma}_t$ where $\hat{\sigma}_t^2$ is the fitted volatility process $\hat{\sigma}_t^2 = \hat{a}_0 + \hat{a}_1 \hat{\sigma}_{t-1}^2 + \hat{b}_1 r_{t-1}^2$. An alternative resampling would to generate normal i.i.d. errors in the resampling. This type of resampling is also called parametric bootstrap. At first sight the volatility clusters in the parametric bootstrap have similar shape as in the plot of the observed logreturns. Figure 2.6 shows local averages $\hat{m}(t)$ over squared logreturns. We have chosen $\hat{m}(t)$ as Nadaraya–Watson estimate $\hat{m}_h(t) = [\sum_{s=1}^N K\{(s-t)/h\} r_s^2] / [\sum_{s=1}^N K\{(s-t)/h\}]$. We used a Gaussian kernel K and bandwidth $h = 5$ days. Figures 2.7–2.9 show the corresponding plots for the three resampling methods. Again the plots for random permutation resampling and nonparametric i.i.d. bootstrap qualitatively differ from the plot for the observed time series (Figs. 2.7 and 2.8). In Fig. 2.9 the GARCH bootstrap shows a qualitatively similar picture as the original logreturns ruling again not out the GARCH(1,1) model.

As a last example we consider plots that measure local and global shape characteristics of the time series. We consider the number of local maxima of the kernel smoother \hat{m}_h as a function of the bandwidth h . We compare this function with the number of local maxima for resamples. Figures 2.10–2.12 show the correspond-

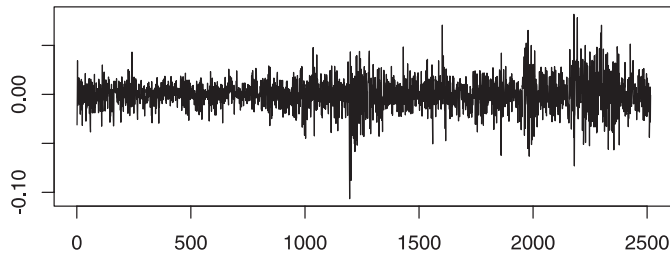


Figure 2.5. GARCH(1,1) bootstrap sample of the data in Fig. 2.2

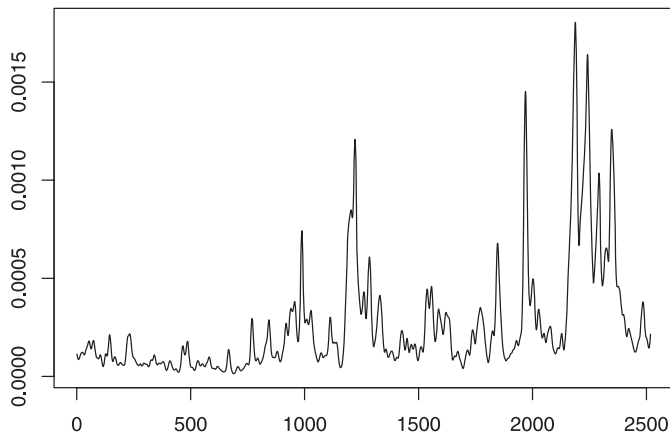


Figure 2.6. Plot of kernel smooth of squared logreturns from the data in Fig. 2.2

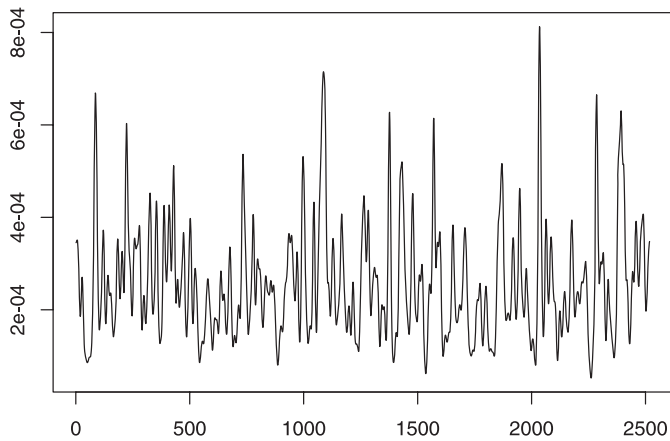


Figure 2.7. Plot of kernel smooth of squared logreturns from the data in Fig. 2.3

ing plots for the permutation resampling, the nonparametric bootstrap and the GARCH(1,1) bootstrap. The plot of the original data set is always compared with the plots for 10 resamples. Again i.i.d. structures are not supported by the resam-

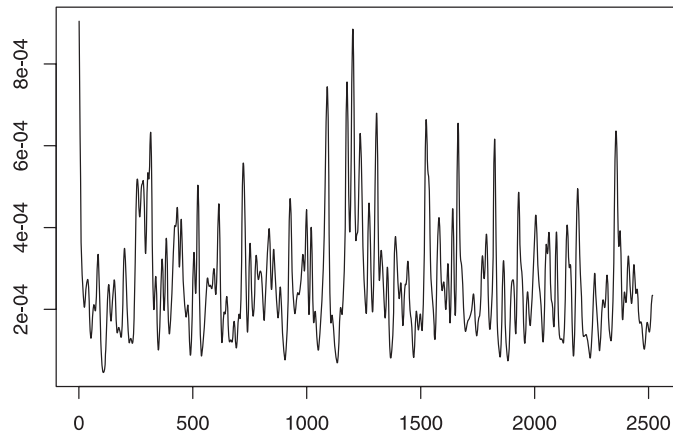


Figure 2.8. Plot of kernel smooth of squared logreturns from the data in Fig. 2.4

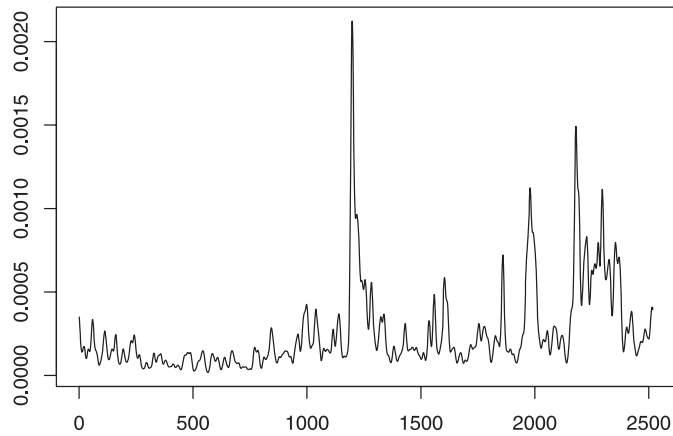


Figure 2.9. Plot of kernel smooth of squared logreturns from the data in Fig. 2.5

pling methods. GARCH(1,1) bootstrap produces plots that are comparable to the original plot.

The last approach could be formalized to a test procedure. This could e.g. be done by constructing uniform resampling confidence bands for the expected number of local maxima. We will discuss resampling tests in the next section. For our last example we would like to mention that there seems to be no simple alternative to resampling. An asymptotic theory for the number of maxima that could be used for asymptotic confidence bands is not available (to our knowledge) and it would be rather complicated. Thus, resampling offers an attractive way out. It could be used for a more data analytic implementation as we have used it here. But it could also be used for getting a formal test procedure.

The first two problems, discussed in Figs. 2.1–2.9, are too complex to be formalized as a testing approach. It is impossible to describe for what differences

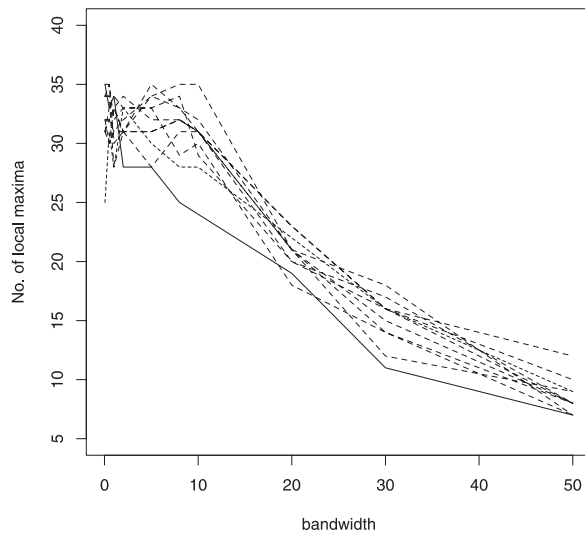


Figure 2.10. Number of local maxima of kernel smoother \hat{m}_h of squared mean-corrected logreturns of DAX data from Fig. 2.1 (*black line*) compared with number of local maxima for 10 random permutation resamples (*dashed lines*)

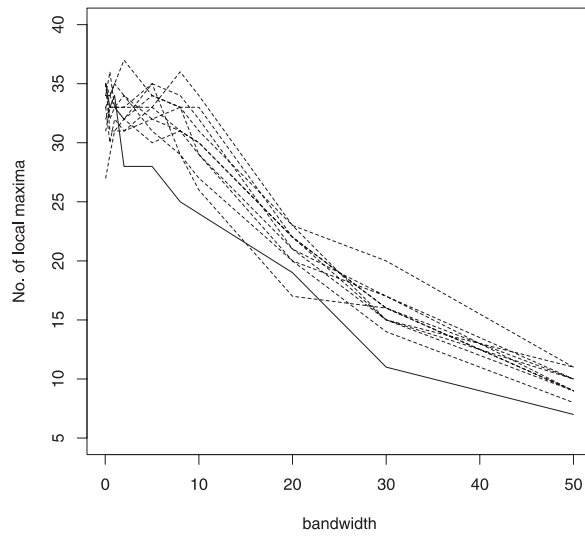


Figure 2.11. Number of local maxima of kernel smoother \hat{m}_h of squared mean-corrected logreturns of DAX data from Fig. 2.1 (*black line*) compared with number of local maxima for 10 nonparametric bootstrap resamples (*dashed lines*)

the human eye is looking in the plots and to summarize the differences in one simple quantity that can be used as a test statistic. The eye is using a battery of “tests” and it is applying the same or similar checks for the resamples.

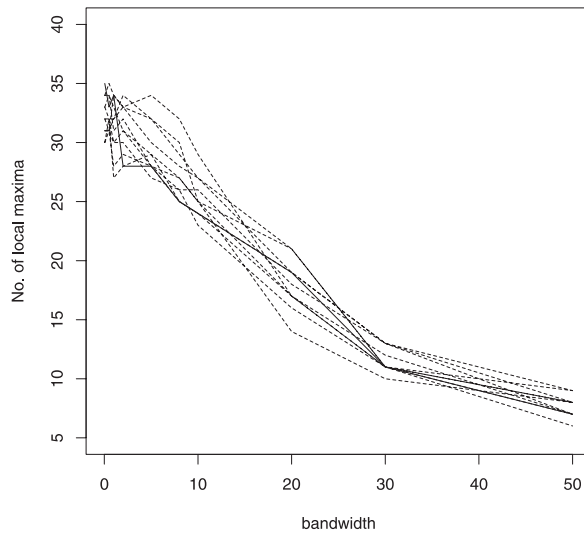


Figure 2.12. Number of local maxima of kernel smoother \hat{m}_h of squared mean-corrected logreturns of DAX data from Fig. 2.1 (*black line*) compared with number of local maxima for GARCH(1,1) bootstrap resamples (*dashed lines*)

Thus, resampling is a good way to judge statistical findings based on the original plots.

2.3

Resampling Tests and Confidence Intervals

In the last section we have pointed out how resampling can offer additional insights in a data analysis. We now want to discuss applications of bootstrap that are more in the tradition of classical statistics. We will introduce resampling approaches for the construction of confidence intervals and of testing procedures. The majority of the huge amount of the bootstrap literature is devoted to these topics. There exist two basic approaches for the construction of confidence regions:

- bootstrapping asymptotic pivots, bootstrap-t intervals
- confidence intervals based on bootstrap percentiles

We will outline both methods below. There also exist two basic approaches for the construction of resampling tests:

- resampling from the hypothesis
- conditional tests

We will discuss testing after confidence intervals.

Approaches based on pivot statistics are classical methods for the construction of confidence sets. In a statistical model $\{P_\theta : \theta \in \Theta\}$ a pivot statistic is a random quantity $Q = Q(\theta, X)$ that depends on the unknown parameter θ and on the observation (vector) X and that has the following property. The distribution of Q under P_θ does not depend on θ . Thus the distribution of Q is known and one can calculate quantiles $q_{1,\alpha}, q_{2,\alpha}$ such that $P_\theta\{q_{1,\alpha} \leq Q(\theta, X) \leq q_{2,\alpha}\} = 1 - \alpha$. Then $C_\alpha = \{\theta \in \Theta : q_{1,\alpha} \leq Q(\theta, X) \leq q_{2,\alpha}\}$ is a confidence set of the unknown parameter θ with coverage probability $P(\theta \in C_\alpha) = 1 - \alpha$. Classical examples are i.i.d. normal observations X_i with mean μ and variance σ^2 . Then $Q = (\bar{X} - \mu)/\hat{\sigma}$ is a pivot statistic. Here \bar{X} is the sample mean and $\hat{\sigma}^2 = (n - 1)^{-1} \sum_{i=1}^n (X_i - \bar{X})^2$ is the sample variance. Then we get, e.g. $C_\alpha = [\bar{X} - n^{-1/2}k_{1-\alpha/2}\hat{\sigma}, \bar{X} + n^{-1/2}k_{1-\alpha/2}\hat{\sigma}]$ is a confidence interval for μ with exact coverage probability $1 - \alpha$. Here $k_{1-\alpha/2}$ is the $1 - \alpha/2$ quantile of the t-distribution with $n - 1$ degrees of freedom.

Pivot statistics only exist in very rare cases. However for a very rich class of settings one can find statistics $Q = Q(\theta, X)$ that have a limiting distribution $\mathcal{L}(\theta)$ that smoothly depends on θ . Such statistics are called asymptotic pivot statistics. If now $q_{1,\alpha}, q_{2,\alpha}$ are chosen such that under $\mathcal{L}(\hat{\theta})$ the interval $[q_{1,\alpha}, q_{2,\alpha}]$ has probability $1 - \alpha$ then we get that $P(\theta \in C_\alpha)$ converges to $1 - \alpha$. Here $\hat{\theta}$ is a consistent estimate of θ and the confidence set C_α is defined as above. A standard example can be easily given if an estimate $\hat{\tau}$ of a (one-dimensional, say) parameter $\tau = \tau(\theta)$ is given that is asymptotically normal. Then $\sqrt{n}(\hat{\tau} - \tau)$ converges in distribution towards a normal limit with mean zero and variance $\sigma^2(\theta)$ depending on the unknown parameter θ . Here $Q = \sqrt{n}(\hat{\tau} - \tau)$ or the studentized version $Q = \sqrt{n}(\hat{\tau} - \tau)/\sigma(\hat{\theta})$ with a consistent estimate $\hat{\theta}$ of θ could be used as asymptotic pivot. Asymptotic pivot confidence intervals are based on the quantiles of the asymptotic distribution \mathcal{L} of Q . The bootstrap idea is to simulate the finite sample distribution $\mathcal{L}_n(\theta)$ of the pivot statistic Q instead of using the asymptotic distribution of Q . This distribution depends on n and on the unknown parameter θ . The bootstrap idea is to estimate the unknown parameter and to plug it in. Then bootstrap quantiles for Q are defined as the (random) quantiles of $\mathcal{L}_n(\hat{\theta})$. For the unstudentized statistic $Q = \sqrt{n}(\hat{\tau} - \tau)$ we get the bootstrap confidence interval $[\hat{\tau} - n^{-1/2}\hat{q}_{2,\alpha}, \hat{\tau} - n^{-1/2}\hat{q}_{1,\alpha}]$ where $\hat{q}_{1,\alpha}$ is the $\alpha/2$ bootstrap quantile and $\hat{q}_{2,\alpha}$ is the $1 - \alpha/2$ bootstrap quantile. This confidence interval has an asymptotic coverage probability equal to $1 - \alpha$. We want to illustrate this approach by the data example of the last section. Suppose we fit a GARCH(1,1) model to the logreturns and we want to have a confidence interval for $\tau = a_1 + b_1$. It is known that a GARCH(1,1) process is covariance stationary if and only if $|\tau| < 1$. For values of τ that approximate 1, one gets a very high persistency of shocks on the process. We now construct a bootstrap confidence interval for τ . We used $Q = \sqrt{n}(\hat{\tau} - \tau)$ as asymptotic pivot statistic. The results are summarized in Table 2.1.

We also applied the GARCH(1,1) bootstrap to the first half and to the second half of our data set. The results are summarized in Table 2.2. The value of $\hat{\tau}$ is quite similar for both halves. The fitted parameter is always contained in the confidence interval based on the other half of the sample. Both confidence intervals have a broad overlap. So there seems no reason to expect different values of τ for the

Table 2.1. Estimate of $a_1 + b_1$ and 90% bootstrap confidence interval using GARCH(1,1) bootstrap (asymptotic pivot method)

$\hat{a}_1 + \hat{b}_1$	Confidence lower bound	Upper bound
0.9919	0.9874	0.9960

two halves of the data. The situation becomes a little bit confused if we compare Table 2.2 with Table 2.1. Both fitted values of τ , the value for the first half and for the second half, are not contained in the confidence interval that is based on the whole sample. This suggests that a GARCH(1,1) model with fixed parameters for the whole sample is not an appropriate model. A model with different values seems to be more realistic. When for the whole time series a GARCH(1,1) model is fitted the change of the parameters in time forces the persistency parameter τ closer to 1 and this effect increases for GARCH fits over longer periods. We do not want to discuss this point further here and refer to Mikosch and Starica (2002) for more details.

Table 2.2. Estimate of $a_1 + b_1$ and 90% bootstrap-t confidence interval using GARCH(1,1) bootstrap for the first half and for the second half of the DAX returns (asymptotic pivot method)

	$\hat{a}_1 + \hat{b}_1$	Confidence lower bound	Upper bound
Using Part I	0.9814	0.9590	0.9976
Using Part II	0.9842	0.9732	0.9888

In Efron (1979) another approach for confidence intervals was suggested. It was supposed to use the bootstrap quantiles of a test statistic directly as bounds of the bootstrap confidence intervals. In our example then the estimate $\hat{\tau}$ has to be calculated repeatedly for bootstrap resamples and the 5% and 95% empirical quantiles are used as lower and upper bound for the bootstrap confidence intervals. It can be easily checked that we then get $[\hat{\tau} + n^{-1/2}\hat{q}_{1,\alpha}, \hat{\tau} + n^{-1/2}\hat{q}_{2,\alpha}]$ as bootstrap confidence interval where the quantiles $\hat{q}_{1,\alpha}$ and $\hat{q}_{2,\alpha}$ are defined as above, see also Efron and Tibshirani (1993). Note that the interval is just reflected around $\hat{\tau}$. The resulting confidence interval for τ is shown in Table 2.3. For asymptotic normal test statistics both bootstrap confidence intervals are asymptotically equivalent. Using higher order Edgeworth expansions it was shown that bootstrap pivot intervals achieve a higher order level accuracy. Modifications of percentile intervals have been proposed that achieve level accuracy of the same order, see Efron and Tibshirani (1993). For a recent discussion on bootstrap confidence intervals see also Efron (2003), Davison, Hinkley and Young (2003). In our data example there is only a minor difference between the two intervals, cf. Tables 2.1 and 2.3. This may be caused by the very large sample size.

The basic idea of bootstrap tests is rather simple. Suppose that for a statistical model $\{P_\theta : \theta \in \Theta\}$ a testing hypothesis $\theta \in \Theta_0 \subset \Theta$ and a test statistic $T(X)$ is given. Then bootstrap is used to calculate critical values for $T(X)$. This can be done by fitting a model on the hypothesis and by generating bootstrap resamples under the fitted hypothesis model. The $1 - \alpha$ quantile of the test statistic in the bootstrap

Table 2.3. Estimate of $a_1 + b_1$ and 90% bootstrap percentile confidence interval using GARCH(1,1) bootstrap

$\widehat{a}_1 + \widehat{b}_1$	Confidence lower bound	Upper bound
0.9919	0.9877	0.9963

samples can be used as critical value. The resulting test is called a bootstrap test. Alternatively, a testing approach can be based on the duality of testing procedures and confidence regions. Each confidence region defines a testing procedure by using the following rule. A hypothesis is rejected if no hypothesis parameter lies in the confidence region. We shortly describe this method for bootstrap confidence intervals based on an asymptotic pivot statistic, say $\sqrt{n}(\widehat{\theta}_n - \theta)$, and the hypothesis $\Theta_0 = (-\infty, \theta_0] \subset \mathbb{R}$. Bootstrap resamples are generated (in the unrestricted model) and are used for estimating the $1 - \alpha$ quantile of $\sqrt{n}(\widehat{\theta}_n - \theta)$ by $\widehat{k}_{1-\alpha}$, say. The bootstrap test rejects the hypothesis, if $\sqrt{n}(\widehat{\theta}_n - \theta_0)$ is larger than $\widehat{k}_{1-\alpha}$. Higher order performance of bootstrap tests has been discussed in Hall (1992). For a discussion of bootstrap tests we also refer to Beran (1988), Beran and Ducharme (1991).

We now compare bootstrap testing with a more classical resampling approach for testing (“conditional tests”). There exist some (important) examples where, for all test statistics, resampling can be used to achieve a correct level on the whole hypothesis for finite samples. Such tests are called similar. For some testing problems resampling tests turn out to be the only way to get similar tests. This situation arises when a statistic is available that is sufficient on the hypothesis $\{P_\theta : \theta \in \Theta_0\}$. Then, by definition of sufficiency, the conditional distribution of the data set given this statistic is fixed on the hypothesis and does not depend on the parameter of the underlying distribution as long as the parameter lies on the hypothesis. Furthermore, because this distribution is unique and thus known, resamples can be drawn from this conditional distribution. The resampling test then has correct level on the whole hypothesis. We will now give a more formal description.

A test $\phi(X)$ for a vector X of observations is called similar if $E_\theta \phi(X) = \alpha$ for all $\theta \in \Theta_0$, where Θ_0 is the set of parameters on the null hypotheses. We suppose that a statistic S is available that is sufficient on the hypothesis. Let $\mathcal{P}_0 = \{P_\theta : \theta \in \Theta_0\}$ be the family of distributions of X on the hypothesis. Then the conditional distribution of X given S does not depend on the underlying parameter $\theta \in \Theta_0$ because S is sufficient. In particular, $E(\phi(X)|S = s)$ does not depend on θ . Then any test satisfying

$$E[\phi(X)|S = s] = \alpha \tag{2.1}$$

is similar on \mathcal{P}_0 . This immediately follows from

$$E[\phi(X)] = EE[\phi(X)|S] = \alpha .$$

A test satisfying (2.1) is said to have Neyman structure with respect to S .

For a given test statistic T similar tests can be constructed by choosing $k_\alpha(S)$ such that

$$P[T > k_\alpha(S)|S = s] = \alpha. \quad (2.2)$$

Here the conditional probability on the left hand side does not depend on θ because S is assumed to be sufficient. We now argue that this is the only way to construct similar tests if the family of distributions of S (for $\theta \in \Theta_0$) is “rich enough”. For such families $E_\theta u(S) = 0$ for all $\theta \in \Theta_0$ for a function u implies $u(s) \equiv 0$. In particular, with $u(s) = P[T > k_\alpha(S)|S = s] - \alpha$ we get that the test $T > k_\alpha(S)$ is similar if $E_\theta u(S) = 0$ for all $\theta \in \Theta_0$. This implies $u(s) \equiv 0$, i.e. (2.2). Thus, given a test statistic T , the only way to construct similar tests is by choosing $k_\alpha(S)$ according to (2.2). The relation between similar tests and tests with Neyman structure belongs to the classical material of mathematical statistics and can be found in text books, e.g. Lehmann (1986).

We will consider two examples of conditional tests. The first one are permutation tests. For a sample of observations $X = (X_1, \dots, X_n)$ the order statistic $S = (X_{(1)}, \dots, X_{(n)})$ containing the ordered sample values $X_{(1)} \leq \dots \leq X_{(n)}$ is sufficient on the hypothesis of i.i.d. observations. Given S , the conditional distribution of X is a random permutation of X_1, \dots, X_n . The resampling scheme is very similar to the nonparametric bootstrap. In the resampling, n pseudo observations are drawn from the original data sample. Now this is done without replacement whereas in the bootstrap scheme this is done with replacement. For a comparison of bootstrap and permutation tests see also Janssen and Pauls (2003). Also for the subsampling (i.e. resampling with a resample size that is smaller than the sample size) both schemes (with and without replacement) have been considered. For a detailed discussion of the subsampling without replacement see Politis, Romano and Wolff (1999).

The second example is a popular approach in the physical literature on nonlinear time series analysis. For odd sample size n a series X_1, \dots, X_n can be written as

$$X_t = \bar{X} + \sqrt{\frac{2\pi}{n}} \sum_{j=1}^{(n-1)/2} 2\sqrt{I_X\left(\frac{2\pi j}{n}\right)} \cos\left(\frac{2\pi j}{n}t + \theta_j\right)$$

with sample mean \bar{X} , periodogram $I_X(\omega) = \frac{1}{2\pi n} \left| \sum_{t=1}^n X_t \exp(-i\omega t) \right|^2$ and phases θ_j . On the hypothesis that X is a circular stationary Gaussian process the statistic $S = (\bar{X}, I_X(2\pi j/n) : j = 1, \dots, (n-1)/2)$ is sufficient. Conditional on S , the phases θ_j are conditional i.i.d. and have a uniform distribution on $[0, 2\pi]$. Resamples with observations

$$X_t^* = \bar{X} + \sqrt{\frac{2\pi}{n}} \sum_{j=1}^{(n-1)/2} 2\sqrt{I_X\left(\frac{2\pi j}{n}\right)} \cos\left(\frac{2\pi j}{n}t + \theta_j^*\right),$$

where θ_j^* are i.i.d. with uniform distribution are called “surrogate data”. They can be used to construct similar tests for the hypothesis of circular stationary Gaussian processes. In the physics literature these tests are applied for testing the hypothesis of stationary Gaussian processes. It is argued that for tests that do not heavily

depend on boundary observations the difference between stationarity and circular stationarity becomes negligible for large data sets. Surrogate data tests are used as first checks if deterministic nonlinear time series models are appropriate for a data set. The relation of surrogate data tests to conditional tests was first observed in Chan (1997). The method of surrogate data was first proposed in Theiler et al. (1992).

We would like to highlight a major difference between bootstrap and conditional tests. Bootstrap tests work if they are based on resampling of an asymptotic pivot statistic. Then the bootstrap critical values stabilize asymptotically and converge against the quantile of the limiting distribution of the test statistic. For conditional tests the situation is quite different. They work for all test statistics. However, not for all test statistics it is guaranteed that the critical value $k_\alpha(S)$ converges to a deterministic limit. In Mammen and Nandi (2004) this is discussed for surrogate data tests. It is shown that also for very large data sets the surrogate data quantile $k_\alpha(S)$ may have a variance of the same order as the test statistic T . Thus the randomness of $k_\alpha(S)$ may change the nature of a test. This is illustrated by a test statistic for kurtosis of the observations that is transformed to a test for circular stationarity.

Bootstrap for Dependent Data

2.4

The Bootstrap for dependent data is a lively research area. A lot of ideas are around and have led to quite different proposals. In this section we do not want to give a detailed overview and description of the different proposals. We only want to sketch the main ideas. Models for dependent data may principally differ from i.i.d. models. For dependent data the data generating process is often not fully specified. Then there exists no unique natural way for resampling. The resampling should be carried out in such a way that the dependence structure should be captured. This can be easily done in case of classical finite-dimensional ARMA models with i.i.d. residuals. In these models the resamples can be generated by fitting the parameters and by using i.i.d. residuals in the resampling. We will discuss the situation when no finite-dimensional model is assumed. For other overviews on the bootstrap for time series analysis, see Bühlmann (2002), Härdle, Horowitz and Kreiss (2003), Politis (2003) and the time series chapter in Davison and Hinkley (1997) and the book Lahiri (2003b). In particular, Härdle, Horowitz and Kreiss (2003) give an overview over the higher order performance of the different resampling schemes.

The most popular bootstrap methods for dependent data are block, sieve, local, wild and Markov bootstrap and subsampling. They all are nonparametric procedures.

The Subsampling

2.4.1

The method that works under a minimal amount of assumptions is the subsampling. It is used to approximate the distribution of an estimate $\hat{\theta}_n$ estimating an

unknown parameter θ . In the subsampling subsamples of consecutive observations of length $l < n$ are taken. These subsamples are drawn randomly from the whole time series. For the subsamples estimates $\hat{\theta}^*$ are calculated. If it is known that for a sequence a_n the statistic $a_n(\hat{\theta}_n - \theta)$ has a limiting distribution then under very weak conditions the conditional distribution of $a_l(\hat{\theta}^* - \hat{\theta}_n)$ has the same limiting distribution. Higher order considerations show that the subsampling has a very poor rate of convergence, see Hall and Jing (1996). It does not even achieve the rate of convergence of a normal approximation. It may be argued that this poor performance is the price for its quite universal applicability. Subsampling has also been used in i.i.d. settings where classical bootstrap does not work. For a detailed discussion of the subsampling see Politis, Romano and Wolf (1999).

2.4.2 The Block Bootstrap

The basic idea of the block bootstrap is closely related to the i.i.d. nonparametric bootstrap. Both procedures are based on drawing observations with replacement. In the block bootstrap however instead of single observations blocks of consecutive observations are drawn. This is done to capture the dependence structure of neighbored observations. Different versions of this idea have been proposed in Hall (1985), Carlstein (1986), Künsch (1989), Liu and Singh (1992b) and Politis and Romano (1994). It has been shown that this approach works for a large class of stationary processes. The blocks of consecutive observations are drawn with replacement from a set of blocks. In the first proposal this was done for a set of nonoverlapping blocks of fixed length l : $\{X_j : j = 1, \dots, l\}, \{X_{l+j} : j = 1, \dots, l\}, \dots$. Later papers proposed to use all (also overlapping) blocks of length l , i.e. the k -th block consists of the observations $\{X_{k-1+j} : j = 1, \dots, l\}$ (Moving block bootstrap). The bootstrap resample is obtained by sampling n/l blocks randomly with replacement and putting them together to a time series of length n . By construction, the bootstrap time series has a nonstationary (conditional) distribution. The resample becomes stationary if the block length l is random and generated from a geometric distribution. This version of the block bootstrap is called the stationary bootstrap and was introduced in Politis and Romano (1994). Recently, Paparoditis and Politis (2001a, 2002a) proposed another modification that uses tapering methods to smooth the effects of boundaries between neighbored blocks. With respect to higher order properties the moving block bootstrap outperforms the version with non overlapping blocks and both achieve a higher order accuracy as the stationary bootstrap (see Hall, Horowitz and Jing (1995), Lahiri (1999a,b, 2003b)).

The block bootstrap has turned out as a very powerful method for dependent data. It does not achieve the accuracy of the bootstrap for i.i.d. data but it outperforms the subsampling. It works reasonably well under very weak conditions on the dependency structure. It has been applied to a very broad range of applications. For the block bootstrap no specific assumption is made on the structure of the data generating process.

We now describe some methods that use more specific assumptions on the dependency structure.

The Sieve Bootstrap

2.4.3

The i.i.d. resampling can also be applied to models of dependent data where the stochastics is driven by i.i.d. innovations. The distribution of the innovations can be estimated by using fitted residuals. In the resampling i.i.d. innovations can be generated by i.i.d. resampling from this fitted distribution. An example is an autoregressive linear model:

$$X_t - \mu_X = \sum_{j=1}^p \varrho_j (X_{t-j} - \mu_X) + \varepsilon_t, \quad t \in \mathbb{Z} \tag{2.3}$$

where $\mu_X = E(X_t)$ is the observation mean and where $\{\varepsilon_t\}$ is a sequence of i.i.d. innovations with $E(\varepsilon_t) = 0$ and ε_t is independent of $\{X_s, s < t\}$. The parameters $\varrho_1, \dots, \varrho_p$ can be estimated by least squares or by using Yule-Walker equations. Residuals can be fitted by putting

$$\bar{\varepsilon}_t = X_t - \hat{\mu}_X - \sum_{j=1}^p \hat{\varrho}_j (X_{t-j} - \hat{\mu}_X),$$

where $\hat{\mu}_X = n^{-1} \sum_{t=1}^n X_t$ and $\hat{\varrho}_1, \dots, \hat{\varrho}_p$ are the fitted parameters. Bootstrap resamples can be generated by

$$X_t^* - \hat{\mu}_X = \sum_{j=1}^p \hat{\varrho}_j (X_{t-j}^* - \hat{\mu}_X) + \varepsilon_t^* \tag{2.4}$$

where ε_t^* are drawn with replacement from the estimated centered residuals $\hat{\varepsilon}_t = \bar{\varepsilon}_t - n^{-1} \sum_{i=1}^n \bar{\varepsilon}_i$. For a study of this bootstrap procedure in model (2.3), see e.g. Franke and Kreiss (1992) and references cited therein.

In a series of papers this approach has been studied for the case that model (2.3) only approximately holds. This is the case if the underlying time series is a stationary linear process, i.e. $\{X_t\}$ has an infinite order autoregressive representation:

$$X_t - \mu_X = \sum_{j=1}^{\infty} \varrho_j (X_{t-j} - \mu_X) + \varepsilon_t, \quad t \in \mathbb{Z}. \tag{2.5}$$

The bootstrap scheme (2.4) has been proposed for this AR(∞) model. In a first step a model (2.3) of finite order p is fitted to the time series. Bootstrap resamples are generated as in (2.4) according to model (2.3). This resampling scheme has been called the sieve bootstrap because the AR(∞) model (2.5) is approximated by an AR(p) model, where, in the asymptotics, p converges to infinity for increasing sample size n . It is argued that this asymptotic approach reflects practical uses of AR models where the order p is selected data adaptively and one is only thinking of the finite order AR model as an approximation to the truth. The Sieve bootstrap and its asymptotic consistency was first considered by Kreiss (1988,1992) and further analyzed by Bühlmann (1997,1998), Bickel and Bühlmann (1999), Paparo-

ditis (1996) and Park (2002). Choi and Hall (2000) showed that under appropriate conditions the sieve bootstrap achieves nearly the rates of convergence of the i.i.d. resampling. In particular, it usually outperforms the block bootstrap. Bühlmann (1997) studied higher order performance of sieve bootstrap variance estimates for the sample mean under assumptions on the decay of the coefficients $\varrho_j \leq cj^{-\nu}$ for constants $c > 0$ and $\nu > 2$.

2.4.4 The Nonparametric Autoregressive Bootstrap

Another residual based bootstrap scheme has been proposed for a nonparametric autoregression model:

$$X_t = m(X_{t-1}, \dots, X_{t-p}) + \sigma(X_{t-1}, \dots, X_{t-q})\varepsilon_t \quad t = 1, 2, \dots \quad (2.6)$$

where $\{\varepsilon_t\}$ is a sequence of i.i.d. error variables with zero mean and unit variance and where m and σ are unknown smooth functions. The functions m and σ can be estimated by nonparametric smoothing estimates \hat{m} and $\hat{\sigma}$. These estimates can be used to fit residuals. In the nonparametric autoregressive bootstrap resamples are generated

$$X_t^* = \hat{m}(X_{t-1}^*, \dots, X_{t-p}^*) + \hat{\sigma}(X_{t-1}^*, \dots, X_{t-q}^*)\varepsilon_t^* \quad t = 1, 2, \dots$$

where \hat{m} and $\hat{\sigma}$ are nonparametric smoothing estimates and where ε_t^* are drawn with replacement from the centered fitted residuals. The choice of the bootstrap autoregression function \hat{m} and of the bootstrap volatility function $\hat{\sigma}^2$ is rather delicate because inappropriate choices can lead to explosive dynamics for the bootstrap time series. The nonparametric autoregressive bootstrap was discussed in Franke, Kreiss and Mammen (2002). They give conditions under which this bootstrap approach is consistent. Franke, Kreiss, Mammen and Neumann (2002) used this bootstrap approach for the construction of uniform confidence bands for the regression function m .

2.4.5 The Regression-type Bootstrap, the Wild Bootstrap and the Local Bootstrap

Franke, Kreiss and Mammen (2002) also consider two other bootstrap procedures for the model (2.6): the regression bootstrap and the wild bootstrap. In the regression bootstrap, a nonparametric regression model is generated with (conditionally) fixed design. We describe this approach for the case of a homoscedastic autoregression model:

$$X_t = m(X_{t-1}, \dots, X_{t-p}) + \varepsilon_t \quad t = 1, 2, \dots \quad (2.7)$$

where again $\{\varepsilon_t\}$ is a sequence of i.i.d. error variables with zero mean and m is an unknown smooth autoregression function. Bootstrap error variables ε_t^* can be generated by drawing with replacement from centered fitted residuals in model (2.7).

In contrast to the autoregression bootstrap the resamples are now generated in a regression model

$$X_t^* = \hat{m}(X_{t-1}, \dots, X_{t-p}) + \varepsilon_t^* \quad t = 1, 2, \dots, \quad (2.8)$$

where \hat{m} is again a nonparametric smoothing estimate of m . The stochastic behavior of the autoregression estimates in model (2.7) is fitted by the bootstrap regression estimates in (2.8). Thus regression of X_t onto $(X_{t-1}, \dots, X_{t-p})$ is mimicked in the bootstrap by regression of X_t^* onto the same covariable $(X_{t-1}, \dots, X_{t-p})$. The regression bootstrap principally differs from the autoregressive bootstrap because no autoregressive scheme is generated in the resampling. Because the original time series is used as covariables in a regression problem the regression bootstrap has the advantage that there is no danger for the bootstrap process to be unstable or to explode. Thus the choice of the bootstrap error distribution and of the estimate \hat{m} is not so crucial as for the autoregression bootstrap. On the other hand the randomness of the covariables is not mimicked in the resampling. This leads to a poorer finite sample performance, see Franke, et al. (2002).

Modifications of the regression bootstrap are the local bootstrap (Paparoditis and Politis, 2000) and the wild bootstrap. The wild bootstrap also uses a regression model with (conditionally) fixed covariables. But it is designed to work also for heteroscedastic errors. It has been first proposed for regression models with independent but not identically distributed error variables, see Wu (1986) and Beran (1986). For nonparametric models it was first proposed in Härdle and Mammen (1993). In the nonparametric autoregression model (2.7) wild bootstrap resamples are generated as in (2.8). But now the error variables ε_t^* are generated as $\varepsilon_t^* = \hat{\varepsilon}_t \eta_t$ where $\hat{\varepsilon}_t$ are centered fitted residuals and where η_1, \dots, η_n are (conditionally) i.i.d. variables with conditional zero mean and conditional unit variance (given the original sample). For achieving higher order accuracy it has also been proposed to use η_t with conditional third moment equal to 1. One could argue that in this resampling scheme the distribution of ε_t is fitted by the conditional distribution of η_t . Then n different distributions are fitted in a model where only n observations are available. This is the reason why in Härdle and Mammen (1993) this approach was called wild bootstrap. For a more detailed discussion of the wild bootstrap, see Liu (1988), Liu and Singh (1992a) and Mammen (1992a,b,1993). The asymptotic analysis of the wild bootstrap and other regression type bootstrap methods in model (2.7) is much simpler than the autoregression bootstrap. In the bootstrap world it only requires mathematical analysis of a nonparametric regression model. Only the discussion of uniform nonparametric confidence bands remains rather complicated because it involves strong approximations of the bootstrap nonparametric regression estimates by Gaussian processes, see Neumann and Kreiss (1998). The wild bootstrap works under quite weak model assumptions. Essentially it is only assumed that the conditional expectation of an observation given the past is a smooth function of the last p observations (for some finite p). Generality has its price. Resampling schemes that use more detailed modeling may achieve a better accuracy. We now consider resampling under the stronger assumption that not on-

ly the mean but also the whole conditional distribution of an observation smoothly depends on the last p observations (for some finite p). Resampling schemes that work under this smooth Markov assumption are the Markov Bootstrap schemes.

2.4.6 The Markov Bootstrap

We discuss the Markov bootstrap for a Markov model of order 1. We will describe two implementations of the Markov bootstrap. For both implementations one has to assume that the conditional distribution of X_{t+1} given X_1, \dots, X_t smoothly depends on X_t . The first version was introduced by Rajarshi (1990). It is based on a nonparametric estimate of the transition density $f(y|x)$ of $X_{t+1} = y$ given $X_t = x$. Using kernel density estimates of the density of X_t and of the joint density of (X_t, X_{t+1}) one can estimate $f(y|x)$ by

$$\widehat{f}(y|x) = \frac{\widehat{f}(x, y)}{\widehat{f}(x)},$$

where

$$\widehat{f}(x, y) = \frac{1}{n-1} \sum_{t=1}^{n-1} K_h(X_t - x) K_g(X_{t+1} - y),$$

$$\widehat{f}(x) = \frac{1}{n} \sum_{t=1}^n K_h(X_t - x)$$

are kernel density estimates with kernel functions $K_r(u) = r^{-1}K(r^{-1}u)$ for bandwidths $r = h, g$. In the bootstrap resampling one starts with an observation X_1^* from the density $\widehat{f}(\cdot)$ and then one iteratively generates X_{t+1}^* by sampling from $\widehat{f}(\cdot|X_t^*)$. Higher order performance of this resampling scheme has been discussed in Horowitz (2003b). It turns out that it achieves faster rates of convergence compared with the block bootstrap. This is in accordance with intuition because the Markov bootstrap requires an additional model assumption, namely the Markov property.

The second version of the Markov bootstrap can be described as a limiting version of the latter for $g \rightarrow 0$. Then in the limiting case the bootstrap process takes values only in the set of observations $\{X_1, \dots, X_n\}$. Given $X_t^* = x$, the next observation X_{t+1}^* is equal to X_s ($2 \leq s \leq n$) with probability $K_h(X_{s-1} - x) / \sum_{r=1}^{n-1} K_h(X_r - x)$. This resampling scheme was introduced in Paparotidis and Politis (2001b, 2002b). Higher order properties are not yet known. It may be expected that it has similar asymptotic properties as the smoothed version of the Markov bootstrap. The unsmoothed version has the advantage that the bootstrap time series is forced to live on the observed values of the original time series. This leads to a more stable dynamic of the bootstrap time series, in particular for smaller sample sizes. Furthermore, for higher dimensional Markov processes the unsmoothed version is based on only d dimensional kernel density smoothing whereas smoothed bootstrap

requires $2d$ dimensional kernel smoothing. Here, d denotes the dimension of the Markov process. Again, one can argue that this leads to a more stable finite sample performance of unsmoothed bootstrap. On the other hand, the smoothed Markov bootstrap takes advantage of smoothness of $f(y|x)$ with respect to y . For larger data sets this may lead to improvements, in case of smooth transition densities.

The Frequency Domain Bootstrap

2.4.7

For the periodogram $I_X(\omega) = \frac{1}{2\pi n} \left| \sum_{t=1}^n X_t \exp(-i\omega t) \right|^2$ it is known that its values for $\omega_j = 2\pi j/n$, $0 < j < n/2$ are asymptotically independent. For the first two moments one gets that for $0 < j, k < n/2, j \neq k$

$$E[I_X(\omega_j)] = f(\omega_j) + o(n^{-1/2}), \tag{2.9}$$

$$\text{Var}[I_X(\omega_j)] = f(\omega_j)^2 + o(1), \tag{2.10}$$

$$\text{Cov}[I_X(\omega_j), I_X(\omega_k)] = n^{-1} f(\omega_j) f(\omega_k) \left[\frac{E[\varepsilon_j^4]}{E[\varepsilon_j^2]^2} - 3 \right] + o(n^{-1}), \tag{2.11}$$

where $f(\omega) = (2\pi)^{-1} \sum_{k=-\infty}^{\infty} \text{Cov}(X_t, X_{t+k}) \exp(-ik\omega)$ is the spectral density of the time series X_t and where $\varepsilon_j = X_j - E[X_j|X_t : t \leq j-1]$ are the innovations of the time series. These expansions hold under some regularity conditions on X_t . In particular, it is needed that X_t is a linear process. Thus approximately, we get that $\eta_j = I_X(\omega_j)/f(\omega_j)$, $0 < j < n/2$ is an i.i.d. sequence. This suggests the following bootstrap scheme, called the frequency domain bootstrap or the periodogram bootstrap.

In this resampling bootstrap values $I_X^*(\omega_j)$ of the periodogram $I_X(\omega_j)$ are generated. The resampling uses two estimates \hat{f} and \tilde{f} of the spectral density. In some implementations these estimates can be chosen identically. The first estimate is used for fitting residuals $\hat{\eta}_j = I_X(\omega_j)/\hat{f}(\omega_j)$. The bootstrap residuals η_1^*, \dots are drawn with replacement from the centered fitted residuals $\hat{\eta}_j/\hat{\eta}$, where $\hat{\eta}$ is the average of $\hat{\eta}_j$ over $0 < j < n/2$. The bootstrap periodogram is then calculated by putting $I_X^*(\omega_j) = \tilde{f}(\omega_j)\eta_j^*$.

The frequency domain bootstrap can be used to estimate the distribution of statistics $n^{-1/2} \sum_{0 < j < n/2} w_j I_X(\omega_j)$. Then the distribution of $n^{-1/2} \sum_{0 < j < n/2} [w_j I_X(\omega_j) - w_j f(\omega_j)]$ is estimated by the conditional distribution of $n^{-1/2} \sum_{0 < j < n/2} [w_j I_X^*(\omega_j) - w_j \tilde{f}(\omega_j)]$. Unfortunately, in general this approach does not work. This can be easily seen by a comparison of the asymptotic variances of the statistics. The original statistic $n^{-1/2} \sum_{0 < j < n/2} w_j I_X(\omega_j)$ has variance that is asymptotically equivalent to

$$n^{-1} \sum w_j^2 f(\omega_j)^2 + \left[\frac{E[\varepsilon_j^4]}{E[\varepsilon_j^2]^2} - 3 \right] \left[n^{-1} \sum w_j f(\omega_j) \right]^2,$$

see (2.9)–(2.11). In the bootstrap world the variance is approximately

$$n^{-1} \sum w_j^2 \tilde{f}(\omega_j)^2 .$$

Thus in general there are differences between the variances that do not vanish asymptotically. The reason is that the term on the right hand side of (2.11) contributes an additional term to the variance for the original time series. This term does not appear in the bootstrap because an i.i.d. resampling is used that produces conditionally uncorrelated $I_X^*(\omega_j)$.

Although the frequency domain bootstrap does not work in general, there exist three important examples where it works. In all three examples the second term in the asymptotic expansion of the variance vanishes. This happens e.g. if the kurtosis of the innovations is equal to zero:

$$\frac{E[\varepsilon_j^4]}{E[\varepsilon_j^2]^2} - 3 = 0 .$$

In particular, this is the case if the innovations have a normal distribution. Another more general example where the bootstrap works is given by statistics where it holds that $n^{-1} \sum w_j f(\omega_j) = o(1)$. A large class of examples for this case are ratio statistics

$$n^{1/2} \frac{\sum_{0 < j < n/2} r_j I_X(\omega_j)}{\sum_{0 < j < n/2} I_X(\omega_j)} .$$

By some Taylor expansion calculus one can see that

$$\begin{aligned} n^{1/2} \left[\frac{\sum_{0 < j < n/2} r_j I_X(\omega_j)}{\sum_{0 < j < n/2} I_X(\omega_j)} - \frac{\sum_{0 < j < n/2} r_j f(\omega_j)}{\sum_{0 < j < n/2} f(\omega_j)} \right] \\ \approx n^{-1/2} \sum_{0 < j < n/2} [w_j I_X(\omega_j) - w_j f(\omega_j)] \end{aligned}$$

with w_j proportional to $r_j - \sum_k r_k f(\omega_k) / \sum_k f(\omega_k)$. Then $\sum_j w_j f(\omega_j) = 0$ and the bootstrap consistently estimates the variance of the ratio statistic. Consistency of the frequency domain bootstrap for ratio statistics has been shown in Dahlhaus and Janas (1996). They also showed that the frequency domain bootstrap achieves higher order accuracy. But for this it is necessary that the third moment of the innovations vanishes. This is a rather restrictive assumption. Examples of ratio statistics are autocorrelation estimates, see Dahlhaus and Janas (1996) where other examples are also given. Modifications of the frequency domain bootstrap have been proposed that work for a larger class of statistics. An example is the proposal of Kreiss and Paparoditis (2003) where ideas of the frequency domain bootstrap are combined with ideas of the sieve bootstrap.

There exists also another example where the frequency domain bootstrap works. Nonparametric smoothing estimates of the spectral density are linear statistics where the weights w_j are now local. For example for kernel smoothing weights $w_j = h^{-1}K[(\omega_j - x)/h]$ with bandwidth h and kernel function K one has $n^{-1} \sum_j w_j^2 f(\omega_j)^2 = O(h^{-1})$. On the other hand, $n^{-1} \sum_j w_j f(\omega_j) = O(1)$ is of lower order. Now, both the variance of the original spectral density estimate and the variance of the bootstrap spectral density estimate have variance that is up to terms of order $o(h)$ is equal to the same quantity $(2\pi)^2 n^{-1} \sum w_j^2 f(\omega_j)^2$. The correlation between $I_X(\omega_j)$ and $I_X(\omega_k)$ for $j \neq k$ (see (2.11)) only contributes to higher order terms. Franke and Härdle (1992) firstly observed this relation and used this fact to show that the frequency domain bootstrap works for nonparametric spectral density estimation. In their approach, both \hat{f} and \tilde{f} are nonparametric kernel smoothing estimates. For \tilde{f} a bandwidth has been chosen that is of larger order than the bandwidth h . Then bootstrap consistently estimates the bias of the spectral density estimate. Similar approaches have been used in bootstrap schemes for other settings of nonparametric curve estimation, see Mammen (2000). For the frequency domain bootstrap for parametric problems one can choose $\hat{f} = \tilde{f}$, see Dahlhaus and Janas (1996).

We now have discussed a large class of resampling schemes for dependent data. They are designed for different assumptions on the dependency structure ranging from quite general stationarity assumptions (subsampling), mixture conditions (block bootstrap), linearity assumptions (sieve bootstrap, frequency domain bootstrap), conditional mean Markov property (wild bootstrap), Markov properties (Markov bootstrap) and autoregressive structure (autoregressive bootstrap). It may be generally conjectured that resampling schemes for more restrictive models are more accurate as long as these more restrictive assumptions really apply. These conjectures are supported by asymptotic results based on higher order Edgeworth expansions. (Although these results should be interpreted with care because of the poor performance of higher order Edgeworth expansions for finite samples, see also the discussion in the introduction.) The situation is also complicated by the fact that in time series analysis typically models are used as approximations to the truth and they are not interpreted as true models. Thus one has to study the much more difficult problem how resampling schemes perform if the underlying assumptions are only approximately fulfilled.

Resampling for dependent data has stimulated very creative ideas and discussions and it had lead to a large range of different approaches. Partially, the resampling structure is quite different from the stochastic structure of the original time series. In the regression bootstrap regression data are used instead of autoregression series. In the sieve bootstrap and in the frequency domain bootstrap models are used that only approximate the original model.

For dependent data the bootstrap has broadened the field of possible statistical applications. The bootstrap offered new ways of implementing statistical procedures and made it possible to treat new types of applied problems by statistical inference.

The discussion of the bootstrap for dependent data is not yet finished. For the comparison of the proposed resampling schemes a complete understanding is still missing and theoretical research is still going on. Applications of time series analysis will also require new approaches. Examples are unit root tests, cointegration analysis and the modeling of financial time series.

References

- Beran, R. (1986). Discussion of Wu, C.F.J.: Jackknife, bootstrap, and other resampling methods in regression analysis (with discussion). *Ann. Statist.*, 14:1295–1298.
- Beran, R. (1988). Prepivoting test statistics: a bootstrap view of asymptotic refinements. *J. Amer. Statist. Assoc.*, 83:687–697.
- Beran, R. (2003). The Impact of the Bootstrap on Statistical Algorithms and Theory. *Statist. Science*, 18:175–184.
- Beran, R. and Ducharme, G. (1991). *Asymptotic theory for bootstrap methods in statistics*, Les Publications CRM, Univ. Montreal.
- Bickel, P.J. and Bühlmann, P. (1999). A new Mixing Notion and Functional Central Limit Theorems for a Sieve Bootstrap in Time Series. *Bernoulli*, 5:413–446.
- Bickel, P. and Freedman, D. (1981). Some asymptotic theory for the bootstrap, *Ann. Statist.*, 9:1196–1217.
- Bickel, P. and Freedman, D. (1983). Bootstrapping regression models with many parameters. In Bickel, P.J., Doksum, K.A. and Hodges, J.C. (eds), *A Festschrift for Erich L. Lehmann*, pp. 28–48, Wadsworth, Belmont.
- Boos, D.D. (2003). Introduction to the bootstrap world. *Statist. Science*, 18:168–174.
- Bose, A. and Chatterjee, S. (2002). Dimension asymptotics for generalized bootstrap in linear regression. *Ann. Inst. Statist. Math.*, 54:367–381.
- Bühlmann, P. (1997). Sieve Bootstrap for Time Series. *Bernoulli*, 3:123–148.
- Bühlmann, P. (1998). Sieve bootstrap for smoothing in nonstationary time series. *Ann. Statist.*, 26:48–83.
- Bühlmann, P. (2002). Bootstraps for time series. *Statist. Science*, 17:52–72.
- Carlstein, E. (1986). The use of subsample methods for estimating the variance of a general statistic from a stationary time series. *Ann. Statist.*, 14:1171–1179.
- Chan, K.S. (1997). On the validity of the method of surrogate data. *Fields Institute Communications*, 11:77–97.
- Choi, E. and Hall, P. (2000). Bootstrap confidence regions computed from autoregressions of arbitrary order. *J. Royal Statist. Soc., Series B*, 62:461–477.
- Dahlhaus, R. and Janas, D. (1996). A frequency domain bootstrap for ratio statistics in time series. *Ann. Statist.*, 24:1934–1963.
- Davison, A.C. and Hinkley, D.V. (1997). *Bootstrap Methods and their Applications*, Cambridge University Press, Cambridge.
- Davison, A.C., Hinkley, D.V. and Young, G.V. (2003). Recent Developments in Bootstrap Methodology. *Statist. Science*, 18:141–157.

- Efron, B. (1979). Bootstrap methods: Another look at jackknife. *Ann. Statist.*, 7:1–26.
- Efron, B. (1982). *The Jackknife, the bootstrap, and other Resampling Plans*. SIAM, Philadelphia.
- Efron, B. (2003). Second Thoughts on the Bootstrap, *Statist. Science*, 18:135–140.
- Efron, B. and Tibshirani, R.J. (1993) *An Introduction to the Bootstrap*, Chapman and Hall, London.
- Franke, J. and Härdle, W. (1992). On bootstrapping kernel spectral estimates. *Ann. Statist.*, 20:121–145.
- Franke, J. and Kreiss, J.-P. (1992). Bootstrapping ARMA-models. *J. Time Series Analysis*, 13:297–317.
- Franke, J., Kreiss, J.-P. and Mammen E. (2002a). Bootstrap of kernel smoothing in nonlinear time series, *Bernoulli*, 8:1–37.
- Franke, J., Kreiss, J.-P., Mammen, E. and Neumann, M.H. (2002b). Properties of the Nonparametric Autoregressive Bootstrap. *J. Time Series Analysis*, 23:555–585.
- Gine, E. (1997). Lectures on some aspects of the bootstrap. In Bernard, P. (ed), *Lectures on Probability Theory and Statistics.*, Berlin: Springer Lecture Notes Math. 1665, pp. 37–151.
- Härdle, W. and Mammen, E. (1991). Bootstrap methods for nonparametric regression. In Roussas, G. (ed), *Nonparametric Functional estimation and Related Topics*, pp. 111–124, Kluwer, Dordrecht.
- Härdle, W. and Mammen, E. (1993). Testing parametric versus nonparametric regression. *Ann. Statist.*, 21:1926–1947.
- Härdle, W., Horowitz, J.L. and Kreiss, J.-P. (2003). Bootstrap methods for time series. *International Statist. Review*, 71:435–459.
- Hall, P. (1985). Resampling a coverage process. *Stoch. Proc. Appl.*, 19:259–269.
- Hall, P. (1992). *The Bootstrap and Edgeworth Expansions*, Springer, New York.
- Hall, P. (2003). A Short Prehistory of the Bootstrap. *Statist. Science*, 18:158–167.
- Hall, P., Horowitz, J.L. and Jing, B.-Y. (1995). On blocking rules for the bootstrap with dependent data. *Biometrika*, 82:561–574.
- Hall, P. and Jing, B.-Y. (1996). On sample reuse methods for dependent data. *J. Royal Statist. Society, Series B*, 58:727–737.
- Horowitz, J.L. (1997). Bootstrap Methods in Econometrics: Theory and Numerical Performance. In Kreps, D.M. and Wallis, K.F. (eds), *Advances in Economics and Econometrics: Theory and Applications*, pp. 188–222, Cambridge University Press.
- Horowitz, J.L. (2001). The Bootstrap. In Heckman, J.J. and Leamer, E.E. (eds), *Handbook of Econometrics*, vol. 5, Chap. 52, 3159–3228, Elsevier Science B.V.
- Horowitz, J.L. (2003a). The bootstrap in econometrics. *Statist. Science*, 18:211–218.
- Horowitz, J.L. (2003b). Bootstrap Methods for Markov Processes. *Econometrica*, 71:1049–1082.
- Inoue, A. and Kilian, L. (2003). The continuity of the limit distribution in the parameter of interest is not essential for the validity of the bootstrap. *Econometric Theory*, 6:944–961.

- Janssen, A. and Pauls, T. (2003). How do bootstrap and permutation tests work? *Annals of Statistics*, 31:768–806.
- Kreiss, J.-P. (1988). *Asymptotic Inference for a Class of Stochastic Processes*, Habilitationsschrift. Faculty of Mathematics, University of Hamburg, Germany.
- Kreiss, J.-P. (1992). Bootstrap procedures for $AR(\infty)$ processes. In Jöckel, K.-H., Rothe, G. and Sendler, W. (eds), *Bootstrapping and Related Techniques*, Lecture Notes in Economics and Mathematical Systems 376, pp. 107–113, Springer, Berlin-Heidelberg-New York.
- Kreiss, J.-P. and Paparoditis, E. (2003). Autoregressive aided periodogram bootstrap for time series. *Ann. Statist.*, 31:1923–1955.
- Künsch, H.R. (1989). The jackknife and the bootstrap for general stationary observations. *Annals of Statistics*, 17:1217–1241.
- Lahiri, S.N. (1999a). Second order optimality of stationary bootstrap. *Statist. Probab. Letters*, 11:335–341.
- Lahiri, S.N. (1999b). Theoretical comparison of block bootstrap methods. *Ann. Statist.*, 27:386–404.
- Lahiri, P. (2003a). On the Impact of Bootstrap in Survey Sampling and Small-Area Estimation. *Statist. Science*, 18:199–210.
- Lahiri, S.N. (2003b). *Resampling Methods for Dependent data*, Springer, New York.
- Lehmann, E.L. (1986). *Testing Statistical Hypotheses*, Springer, New York.
- Lele, S.R. (2003). Impact of Bootstrap on the Estimating Functions. *Statist. Science*, 18:185–190.
- Liu, R.Y. (1988). Bootstrap procedures under some non i.i.d. models. *Ann. Statist.*, 16:1696–1708.
- Liu, R.Y. and Singh, K. (1992a). Efficiency and robustness in resampling. *Ann. Statist.*, 20:370–384.
- Liu, R.Y. and Singh, K. (1992b). Moving blocks jackknife and bootstrap capture weak dependence. In Lepage, R. and Billard, L. (eds), *Exploring the Limits of the Bootstrap*, pp. 225–248, Wiley, New York.
- Mammen, E. (1989). Asymptotics with increasing dimension for robust regression with applications to the bootstrap. *Ann. Statist.*, 17:382–400.
- Mammen, E. (1992a). Bootstrap, wild bootstrap, and asymptotic normality. *Probab. Theory Related Fields*, 93:439–455.
- Mammen, E. (1992b). *When does bootstrap work? Asymptotic results and simulations*, Springer Lecture Notes in Statistics 77, Springer, Heidelberg, Berlin.
- Mammen, E. (1993). Bootstrap and wild bootstrap for high-dimensional linear models. *Ann. Statist.*, 21:2555–285.
- Mammen, E. (2000). Resampling methods for nonparametric regression. In Schimek, M.G. (ed), *Smoothing and Regression: Approaches, Computation and Application*, Wiley, New York.
- Mammen, E. and Nandi, S. (2004). Change of the nature of a test when surrogate data are applied. (To appear in *Physical Review E*)
- Mikosch, T. and Starica, C. (2002). Nonstationarities in financial time series, the long range dependence and the IGARCH effects. (Preprint)

- Neumann, M. and Kreiss, J.-P. (1988). Regression-type inference in nonparametric autoregression. *Ann. Statist.*, 26:1570–1613.
- Paparoditis, E. (1996). Bootstrapping autoregressive and moving average parameter estimates of infinite order vector autoregressive processes. *J. Multivariate Anal.*, 57:277–296.
- Paparoditis, E. and Politis, D.N. (2000). The local bootstrap for kernel estimators under general dependence conditions. *Ann. Inst. Statist. Math.*, 52:139–159.
- Paparoditis, E. and Politis, D.N. (2001a). Tapered block bootstrap. *Biometrika*, 88:1105–1119.
- Paparoditis, E. and Politis, D.N. (2001b). A Markovian local resampling scheme for nonparametric estimators in time series analysis. *Econometric Theory*, 17:540–566.
- Paparoditis, E. and Politis, D.N. (2002a). The tapered block bootstrap for general statistics from stationary sequences. *The Econometrics Journal*, 5:131–148.
- Paparoditis, E. and Politis, D.N. (2002b). The local bootstrap for Markov processes. *J. Statist. Planning Inference*, 108:301–328.
- Park, J.Y. (2002). An invariance principle for sieve bootstrap in time series, *Econometric Theory*, 18:469–490.
- Politis, D.N. (2003). The Impact of Bootstrap Methods on Time Series Analysis. *Statist. Science*, 18:219–230.
- Politis, D.N. and Romano, J.P. (1994). The Stationary Bootstrap. *J. Amer. Statist. Assoc.*, 89:1303–1313.
- Politis, D.N., Romano, J.P. and Wolf, M. (1999). *Subsampling*, Springer, New York.
- Rajarshi, M.B. (1990). Bootstrap in Markovsequences based on estimates of transition density. *Ann. Inst. Statist. Math.*, 42:253–268.
- Shao, J. (1996). Resampling methods in sample surveys (with discussions). *Statistics*, 27:203–254.
- Shao, J. (2003). Impact of the Bootstrap on Sample Surveys. *Statist. Science*, 18:191–198.
- Shao, J. and Tu, T. (1995). *The Jackknife and Bootstrap*, Springer, New York.
- Theiler, J., Eubank, S., Longtin, A., Galdrikan, B. and Farmer, J.D. (1992). Testing for nonlinearity in time series: the method of surrogate data. *Physica D*, 58:77–94.
- Wu, C.F.J. (1986). Jackknife, bootstrap, and other resampling methods in regression analysis (with discussion). *Ann. Statist.*, 14:1261–1295.

Design and Analysis of Monte Carlo Experiments

III.3

Jack P.C. Kleijnen

3.1	<i>Introduction</i>	498
3.2	<i>Simulation Techniques in Computational Statistics</i>	499
3.3	<i>Black-Box Metamodels of Simulation Models</i>	501
3.4	<i>Designs for Linear Regression Models</i>	502
	Simple Regression Models for Simulations with a Single Factor	502
	Simple Regression Models for Simulation Models with Multiple Factors	504
	Fractional Factorial Designs and Other Incomplete Designs	507
	Designs for Simulations with Too Many Factors	509
3.5	<i>Kriging</i>	510
	Kriging Basics	510
	Designs for Kriging	512
3.6	<i>Conclusions</i>	513

Introduction

By definition, computer simulation (or Monte Carlo) models are not solved by mathematical analysis (for example, differential calculus), but are used for numerical experimentation. These experiments are meant to answer questions of interest about the real world; i.e., the experimenters may use their simulation model to answer *what if* questions – this is also called *sensitivity analysis*. Sensitivity analysis – guided by the statistical theory on *design of experiments* (DOE) – is the focus of this chapter. Sensitivity analysis may further serve validation, optimization, and risk (or uncertainty) analysis for finding robust solutions; see Kleijnen (1998), Kleijnen et al. (2003a,b). Note that optimization is also discussed at length in Chap. II.6 by Spall.

Though I assume that the reader is familiar with basic simulation, I shall summarize a simple Monte Carlo example (based on the well-known Student *t*-statistic) in Sect. 3.2. This example further illustrates bootstrap and variance reduction techniques

Further, I assume that the reader's familiarity with DOE is restricted to elementary DOE. In this chapter, I summarize classic DOE, and extend it to newer methods (for example, DOE for interpolation using Kriging; Kriging is named after the South-African mining engineer D.G. Krige).

Traditionally, 'the shoemaker's children go barefoot'; i.e., users of computational statistics ignore statistical issues – such as sensitivity analysis – of their simulation results. Nevertheless, they should address *tactical* issues – the number of (macro)replicates, variance reduction techniques – and *strategic* issues – situations to be simulated and the sensitivity analysis of the resulting data. Both types of issues are addressed in this chapter.

Note the following terminology. DOE speaks of 'factors' with 'levels', whereas simulation analysts may speak of 'inputs' or 'parameters' with 'values'. DOE talks about 'design points' or 'runs', whereas simulationists may talk about 'situations', 'cases', or 'scenarios'.

Classic DOE methods for real, non-simulated systems were developed for agricultural experiments in the 1930s, and – since the 1950s – for experiments in engineering, psychology, etc. (Classic designs include fractional factorials, as we shall see.) In those real systems it is impractical to experiment with 'many' factors; $k = 10$ factors seems a maximum. Moreover, it is then hard to experiment with factors that have more than 'a few' values; five values per factor seems a maximum. Finally, these experiments are run in 'one shot' – for example, in one growing season – and not sequentially. In simulation, however, these limitations do not hold!

Two textbooks on classic DOE for simulation are Kleijnen (1975, 1987). An update is Kleijnen (1998). A bird-eye's view of DOE in simulation is Kleijnen et al. (2003a), which covers a wider area than this review.

Note further the following terminology. I speak of the *Monte Carlo* method whenever (pseudo)random numbers are used; for example, I apply the Monte Car-

lo method to estimate the behavior of the t statistic in case of non-normality, in Sect. 3.2 (the Monte Carlo method may also be used to estimate multiple integrals, which is a deterministic problem, outside the scope of this handbook). I use the term *simulation* whenever the analysts compute the output of a dynamic model; i.e., the analysts do not use calculus to find the solution of a set of differential or difference equations. The dynamic model may be either stochastic or deterministic. Stochastic simulation uses the Monte Carlo method; it is often applied in telecommunications and logistics. Deterministic simulation is often applied in computer-aided engineering (CAE). Finally, I use the term *metamodel* for models that approximate – or model – the input/output (I/O) behavior of the underlying simulation model; for example, a polynomial regression model is a popular metamodel (as we shall see). Metamodels are used – consciously or not – to design and analyze experiments with simulation models. In the simulation literature, metamodels are also called response surfaces, emulators, etc.

The remainder of this chapter is organized as follows. Section 3.2 presents a simple Monte Carlo experiment with Student's t statistic, including bootstrapping and variance reduction techniques. Section 3.3 discusses the black box approach to simulation experiments, and corresponding metamodels – especially, polynomial and Kriging models. Section 3.4 starts with simple regression models with a single factor; proceeds with designs for multiple factors including designs for first-order and second-order polynomial models, and concludes with screening designs for hundreds of factors. Section 3.5 introduces Kriging interpolation, which has hardly been applied in random simulation – but has already established a track record in deterministic simulation and spatial statistics. Kriging often uses space-filling designs, such as Latin hypercube sampling (LHS). Section 3.6 gives conclusions and further research topics.

Simulation Techniques in Computational Statistics

3.2

Consider the well-known definition of the t statistic with $n - 1$ degrees of freedom:

$$t_{n-1} = \frac{\bar{x} - \mu}{s_x / \sqrt{n}}, \quad (3.1)$$

where the x_i ($i = 1, \dots, n$) are assumed to be normally (Gaussian), independently, and identically distributed (NIID) with mean μ and variance σ^2 :

$$x_i \in \text{NIID}(\mu, \sigma) (i = 1, \dots, n). \quad (3.2)$$

Nearly 100 years ago, Gossett used a kind of Monte Carlo experiment (without using computers, since they were not yet invented), before he analytically derived the density function of this statistic (and published his results under the pseudo-

nym of Student). So, he sampled n values x_i (from an urn) satisfying (3.2), and computed the corresponding value for the statistic defined by (3.1). This experiment he repeated (say) m times, so that he could compute the estimated density function (EDF) – also called the empirical cumulative distribution function (ECDF) – of the statistic. (Inspired by these empirical results, he did his famous analysis.)

Let us imitate his experiment, in the following simulation experiment (this procedure is certainly not the most efficient computer program).

1. Read the simulation inputs: μ (mean), σ^2 (variance), n (sample size), m (number of macro-replicates, used in step 4).
2. Take n samples $x_i \in \text{NIID}(\mu, \sigma)$ (see (3.2)) and Chap. II.2 by L'Ecuyer).
3. Compute the statistic t_{n-1} (see (3.1)).
4. Repeat steps 2 and 3 m times.
5. Sort the m values of t_{n-1} .
6. Compute the EDF from the results in step 5.

To verify this simulation program, we may compare the result (namely the EDF) with the results that are tabulated for Student's density function; for example, does our EDF give a 90% quantile that is not significantly different from the tabulated value (say) $t_{n-1;0.90}$. Next we may proceed to the following more interesting application.

We may drop the classic assumption formulated in (3.2), and experiment with *non-normal* distributions. It is easy to sample from such distributions (see again Chap. II.2). However, we are now confronted with several so-called *strategic* choices (also see step 1 above): Which type of distribution should be selected (lognormal, exponential, etc.); which parameter values for that distribution type (mean and variance for the lognormal, etc.), which sample size (for asymptotic, 'large' n , the t distribution is known to be a good approximation for our EDF).

Besides these choices, we must face some *tactical* issues: Which number of macro-replicates m gives a good EDF; can we use special *variance reducing techniques* (VRTs) – such as common random numbers and importance sampling – to reduce the variability of the EDF? We explain these techniques briefly, as follows.

Common random numbers (CRN) mean that the analysts use the same (pseudo)random numbers (PRN) – symbol r – when estimating the effects of different strategic choices. For example, CRN are used when comparing the estimated quantiles $\hat{t}_{n-1;0.90}$ for various distribution types. Obviously, CRN reduces the variance of estimated differences, provided CRN creates positive correlations between the estimators $\hat{t}_{n-1;0.90}$ being compared.

Antithetic variates (AV) mean that the analysts use the complements $(1 - r)$ of the PRN (r) in two 'companion' macro-replicates. Obviously, AV reduces the variance of the estimator averaged over these two replicates, provided AV creates negative correlation between the two estimators resulting from the two replicates.

Importance sampling (IS) is used when the analysts wish to estimate a *rare* event, such as the probability of the Student statistic exceeding the 99.999% quantile. IS increases that probability (for example, by sampling from a distribution with a fatter tail) – and later on, IS corrects for this distortion of the input dis-

tribution (through the likelihood ratio). IS is not so simple as CRN and AV – but without IS too much computer time may be needed. See Glasserman et al. (2000).

There are many more VRTs. Both CRN and AV are intuitively attractive and easy to implement, but the most popular one is CRN. The most useful VRT may be IS. In practice, the other VRTs often do not reduce the variance drastically so many users prefer to spend more computer time instead of applying VRTs. (VRTs are a great topic for doctoral research!) For more details on VRTs, I refer to Kleijnen and Rubinstein (2001).

Finally, the density function of the sample data x_i may not be an academic problem: Suppose a very limited set of historical data is given, and we must analyze these data while we know that these data do not satisfy the classic assumption formulated in (3.2). Then *bootstrapping* may help, as follows (also remember the six steps above).

1. Read the bootstrap sample size B (usual symbol in bootstrapping, comparable with m – number of macro-replicates – in step 1 above).
2. Take n samples with replacement from the original sample x_i ; this sampling gives x_i^* (the superscript * denotes bootstrapped values, to be distinguished from the original values).
3. From these x_i^* compute the statistic t_{n-1}^* (see (3.1)).
4. Repeat steps 2 and 3 B times.
5. Sort the B values of t_{n-1}^* .
6. Compute the EDF from the results in step 5.

In summary, bootstrapping is just a Monte Carlo experiment – using resampling with replacement of a given data set. (There is also a parametric bootstrap, which comes even closer to our simulation of Gossett’s original experiment.) Bootstrapping is further discussed in Efron and Tibshirani (1993) and in Chap. III.2 (by Mammen).

Black-Box Metamodels of Simulation Models

3.3

DOE treats the simulation model as a *black box*; i.e., only the inputs and outputs are observed and analyzed. For example, in the simulation of the t statistic (in Sect. 3.2) the simulation inputs (listed in Step 1) are μ (mean), σ^2 (variance), n (sample size), and m (number of macro-replicates); this m is probably a tactical factor that is not of interest to the user. Suppose the user is interested in the 90% quantile of the distribution function of the statistic in case of nonnormality. A *black box* representation of this example is:

$$t_{n-1;0.90} = t(\mu, \sigma, n, r_0), \quad (3.3)$$

where $t(\cdot)$ denotes the mathematical function implicitly defined by the simulation program (outlined in steps 1 through 6 in Sect. 3.2); μ and σ now denote the parameters of the nonnormal distribution of the input x_i (for example, μ denotes how many exponential distributions with parameter $\sigma = \lambda$ are summed to form an Erlang distribution); r_0 denotes the seed of the pseudorandom numbers.

One possible *metamodel* of the black box model in (3.3) is a Taylor series approximation – cut off after the first-order effects of the three factors, μ, σ, n :

$$y = \beta_0 + \beta_1\mu + \beta_2\sigma + \beta_3n + e, \quad (3.4)$$

where y is the metamodel predictor of the simulation output $t_{n-1;0.90}$ in (3.3); $\beta^T = (\beta_0, \beta_1, \beta_2, \beta_3)$ denotes the parameters of the metamodel in (3.4), and e is the noise – which includes both *lack of fit* of the metamodel and *intrinsic noise* caused by the pseudorandom numbers.

Besides the metamodel specified in (3.4), there are many alternative metamodels. For example, taking the logarithm of the inputs and outputs in (3.4) makes the first-order polynomial approximate relative changes; i.e., the parameters β_1, β_2 , and β_3 become elasticity coefficients.

There are many – more complex – types of metamodels. Examples are Kriging models, neural nets, radial basis functions, splines, support vector regression, and wavelets; see the various chapters in Part III – especially Chaps. III.5 (by Loader), III.7 (Müller), III.8 (Cizek), and III.15 (Laskov and Müller) – and also Clarke, Griebisch, and Simpson (2003) and Antoniadis and Pham (1998). I, however, will focus on two types that have established a track record in simulation:

- linear regression models (see Sect. 3.4)
- Kriging (see Sect. 3.5).

To estimate the parameters of whatever metamodel, the analysts must *experiment* with the simulation model; i.e., they must change the inputs (or factors) of the simulation, run the simulation, and analyze the resulting input/output data. This experimentation is the topic of the next sections.

3.4

Designs for Linear Regression Models

Simple Regression Models for Simulations with a Single Factor

3.4.1

I start with the simplest metamodel, namely a first-order polynomial with a single factor. An example is the ‘Student’ simulation in Sect. 3.2, where I now assume that we are interested only in the power so y in (3.4) now denotes the type II error predicted through the regression model. I further assume a single factor (say) $x = \sigma/n$ (‘relative’ variability; i.e., absolute variability corrected for sample size); see (3.4). Elementary mathematics proves that – to fit a straight line – it suffices

to have two input/output observations; see ‘local area 1’ in Fig. 3.1. It is simple to prove that the ‘best’ estimators of the regression parameters in (3.4) result if those two values are as far apart as ‘possible’.

In practice, the analysts do not know over which *experimental area* a first-order polynomial is a ‘valid’ model. This validity depends on the goals of the simulation study; see Kleijnen and Sargent (2000).

So the analysts may start with a *local area*, and simulate the two (locally) extreme input values. Let’s denote these two extreme values of the ‘coded’ variable x by -1 and $+1$, which implies the following *standardization* of the original variable z :

$$x = \frac{z - \bar{z}}{(z_{\max} - z_{\min})/2}, \tag{3.5}$$

where \bar{z} denotes the average value of the relative variability $z = \sigma/n$ in the (local) experiment.

The Taylor series argument implies that – as the experimental area gets bigger (see ‘local area 2’ in Fig. 3.1) – a better metamodel may be a second-order polynomial:

$$y = \beta_0 + \beta_1x + \beta_2x^2 + e. \tag{3.6}$$

Obviously, estimation of the three parameters in (3.6) requires the simulation of at least three input values. Indeed, DOE provides designs with three values per factor; for example, 3^k designs. However, most publications on the application of DOE in simulation discuss *Central Composite Designs* (CCD), which have five values per factor; see Kleijnen (1975).

I emphasize that the second-order polynomial in (3.6) is nonlinear in x (the regression variable), but *linear* in β (the regression parameters or factor effects to be estimated). Consequently, such a polynomial is a type of *linear regression* model (also see Chap. II.8).

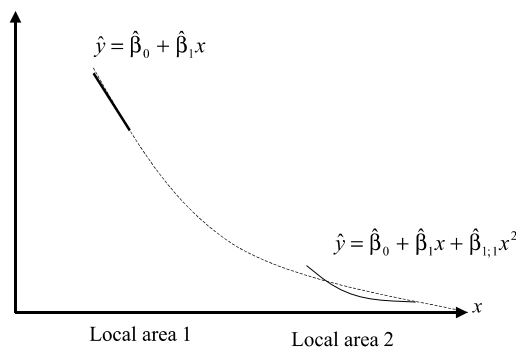


Figure 3.1. Two simple polynomial regression models with predictor \hat{y} for the output of a simulation with a single factor x

Finally, when the experimental area covers the *whole* area in which the simulation model is valid (see again Fig. 3.1), then other *global* metamodels become relevant. For example, Kleijnen and Van Beers (2003a) find that *Kriging* (discussed in Sect. 3.5) outperforms second-order polynomial fitting.

Note that Zeigler, Praehofer, and Kim (2000) call the experimental area the ‘experimental frame’. I call it the domain of admissible scenarios, given the goals of the simulation study.

I conclude that *lessons* learned from the simple example in Fig. 3.1, are:

1. The analysts should decide whether they want to experiment *locally* or *globally*.
2. Given that decision, they should select a specific *metamodel type* (low-order polynomial, Kriging, spline, etc.); also see Chaps. III.5, III.7, and III.8.

Simple Regression Models for Simulation Models with Multiple Factors

3.4.2

Let’s now consider a regression model with k factors; for example, $k = 2$. The design that is still most popular – even though it is inferior – *changes one factor at a time*. For $k = 2$ such a design is shown in Fig. 3.2 and Table 3.1; in this table the factor values over the various factor combinations are shown in the columns denoted by x_1 and x_2 ; the ‘dummy’ column x_0 corresponds with the polynomial intercept $\hat{\beta}_0$ in (3.4). In this design the analysts usually start with the ‘base’ scenario, denoted

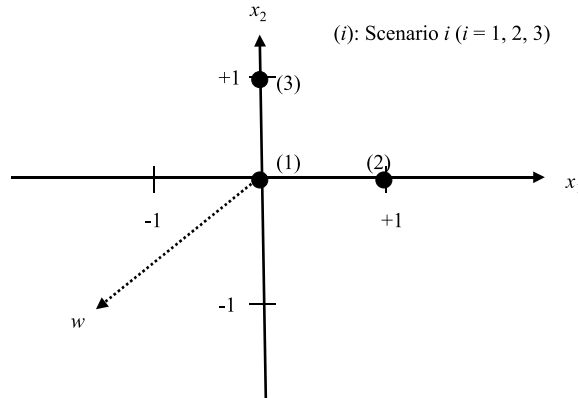


Figure 3.2. One-factor-at-a-time design for two factors x_1 and x_2 , with output w

Table 3.1. A one-factor-at-a-time design for two factors, and possible regression variables

Scenario	x_0	x_1	x_2	x_1x_2
1	1	0	0	0
2	1	1	0	0
3	1	0	1	0

Table 3.2. The 2^3 design and possible regression variables

Scenario	0	1	2	3	1.2	1.3	2.3	1.2.3
1	+	-	-	-	+	+	+	-
2	+	+	-	-	-	-	+	+
3	+	-	+	-	-	+	-	+
4	+	+	+	-	+	-	-	-
5	+	-	-	+	+	-	-	+
6	+	+	-	+	-	+	-	-
7	+	-	+	+	-	-	+	-
8	+	+	+	+	+	+	+	+

by the factor combination (0, 0); see scenario 1 in the table. Next they run the two scenarios (1, 0) and (0, 1); see the scenarios 2 and 3 in the table..

In a one-factor-at-a-time design, the analysts cannot estimate the *interaction* between the two factors. Indeed, Table 3.1 shows that the estimated interaction (say) $\beta_{1,2}$ is *confounded* with the estimated intercept $\hat{\beta}_0$; i.e., the columns for the corresponding regression variables are linearly dependent. (Confounding remains when the base values are denoted not by zero but by one; then these two columns become identical.)

In practice, analysts often study each factor at *three levels* (which may be denoted by $-1, 0, +1$) in their one-at-a-time design. However, two levels suffice to estimate the parameters of a first-order polynomial (see again Sect. 3.4.1).

To enable the estimation of interactions, the analysts must change factors *simultaneously*. An interesting problem arises if k increases from two to three. Then Fig. 3.2 becomes Fig. 3.3, which does not show the output (w), since it would require a fourth dimension (instead x_3 replaces w); the asterisks are explained in

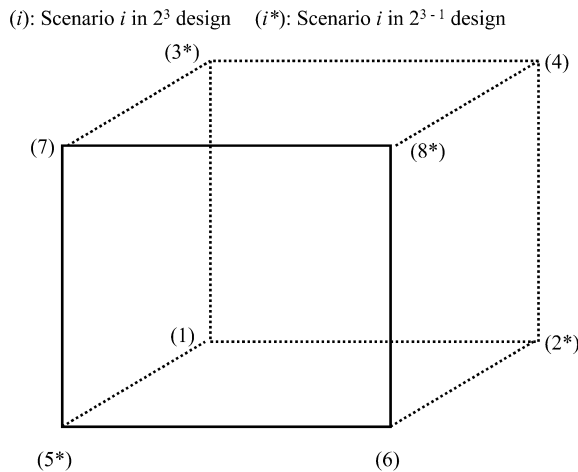


Figure 3.3. The 2^3 design

Sect. 3.4.3. And Table 3.1 becomes Table 3.2. The latter table shows the 2^3 factorial design; i.e., in the experiment each of the three factors has two values and all their combinations of values are simulated. To simplify the notation, the table shows only the signs of the factor values, so $-$ means -1 and $+$ means $+1$. The table further shows possible regression variables, using the symbols '0' through '1.2.3' – to denote the indexes of the regression variables x_0 (the dummy, always equal to 1) through $x_1x_2x_3$ (third-order interaction). Further, I point out that each column is *balanced*; i.e., each column has four plusses and four minuses – except for the dummy column.

The 2^3 design enables the estimation of all eight parameters of the following regression model, which is a third-order polynomial that is *incomplete*; i.e., some parameters are assumed zero:

$$y = \beta_0 + \sum_{j=1}^3 \beta_j x_j + \sum_{j=1}^2 \sum_{j'>j}^3 \beta_{jj'} x_j x_{j'} + \beta_{1;2;3} x_1 x_2 x_3 + e. \quad (3.7)$$

Indeed, the 2^3 design implies a matrix of regression variables \mathbf{X} that is *orthogonal*:

$$\mathbf{X}^T \mathbf{X} = n\mathbf{I}, \quad (3.8)$$

where n denotes the number of scenarios simulated; $n = 8$ in Table 3.2. Hence the *ordinary least squares* (OLS) estimator

$$\widehat{\boldsymbol{\beta}} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{w} \quad (3.9)$$

simplifies for the 2^3 design – which is orthogonal so (3.8) holds – to $\widehat{\boldsymbol{\beta}} = \mathbf{X}^T \mathbf{w} / 8$.

The *covariance matrix* of the (linear) OLS estimator given by (3.9) is

$$\text{cov}(\widehat{\boldsymbol{\beta}}) = [(\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T] \text{cov}(\mathbf{w}) [(\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T]^T. \quad (3.10)$$

In case of *white noise*; i.e.,

$$\text{cov}(\mathbf{w}) = \sigma^2 \mathbf{I}, \quad (3.11)$$

(3.10) reduces to the well-known formula

$$\text{cov}(\widehat{\boldsymbol{\beta}}) = \sigma^2 (\mathbf{X}^T \mathbf{X})^{-1}. \quad (3.12)$$

However, I claim that in practice this white noise assumption does not hold:

1. The output variances change as the input changes so the assumed common variance σ^2 in (3.11) does not hold. This is called *variance heterogeneity*. (Well-known examples are Monte Carlo studies of the type I and type II errors, which give binomial variables so the estimated variances are $y(1-y)/m$; also see Sect. 3.2)
2. Often the analysts use *common random numbers* (see CRN in Sect. 3.2), so the assumed diagonality of the matrix in (3.11) does not hold.

Therefore I conclude that the analysts should choose between the following two options.

1. Continue to apply the OLS *point* estimator (3.9), but use the *covariance* formula (3.10) instead of (3.12)
2. Switch from OLS to *Generalized Least Squares* (GLS) with $\text{cov}(\mathbf{w})$ estimated from $m > n$ replications (so the estimated covariance matrix is not singular); for details see Kleijnen (1992, 1998).

The variances of the estimated regression parameters – which are on the main diagonal of $\text{cov}(\hat{\boldsymbol{\beta}})$ in (3.10) – can be used to test statistically whether some factors have zero effects. However, I emphasize that a significant factor may be unimportant – practically speaking. If the factors are scaled between -1 and $+1$ (see the transformation in (3.5)), then the estimated effects quantify the *order of importance*. For example, in a first-order polynomial regression model the factor estimated to be the most important factor is the one with the highest absolute value for its estimated effect. See Bettonvil and Kleijnen (1990).

Fractional Factorial Designs and Other Incomplete Designs

3.4.3

The incomplete third-order polynomial in (3.7) included a third-order effect, namely $\beta_{1,2,3}$. Standard DOE textbooks include the definition and estimation of such high-order interactions. However, the following claims may be made:

1. High-order effects are hard to interpret
2. These effects often have negligible magnitudes.

Claim # 1 seems obvious. If claim #2 holds, then the analysts may simulate fewer scenarios than specified by a full factorial (such as the 2^3 design). For example, if $\beta_{1,2,3}$ is indeed zero, then a 2^{3-1} fractional factorial design suffices. A possible 2^{3-1} design is shown in Table 3.2, deleting the four rows (scenarios) that have a minus sign in the 1.2.3 column (i.e., delete the rows 1, 4, 6, 7). In other words, only a *fraction* – namely 2^{-1} of the 2^3 full factorial design – is simulated. This design corresponds with the points denoted by the symbol * in Fig. 3.3. Note that this figure has the following geometrically property: each scenario corresponds with a vertex that cannot be reached via a single edge of the cube.

In this 2^{3-1} design two columns are identical, namely the 1.2.3 column (with four plusses) and the dummy column. Hence, the corresponding two effects are confounded – but the high-order interaction $\beta_{1,2,3}$ is assumed zero, so this confounding can be ignored!

Sometimes a *first-order polynomial* suffices. For example, in the (sequential) optimization of black-box simulation models the analysts may use a first-order polynomial to estimate the local gradient; see Angün et al. (2002). Then it suffices to take a 2^{k-p} design with the biggest p value that makes the following condition hold: $2^{k-p} > k$. An example is: $k = 7$ and $p = 4$ so only 8 scenarios are simulated;

Resolution IV designs enable unbiased estimators of first-order effects – even if two-factor interactions are important. These designs require double the number of scenarios required by resolution III designs; i.e., after simulating the scenarios of the resolution III design, the analysts simulate the *mirror scenarios*; i.e., multiply by -1 the factor values in the original scenarios.

Resolution V designs enable unbiased estimators of first-order effects plus all two-factor interactions. To this class belong certain 2^{k-p} designs with small enough p values. These designs often require rather many scenarios to be simulated. Fortunately, there are also *saturated* designs; i.e., designs with the minimum number of scenarios that still allow unbiased estimators of the regression parameters. Saturated designs are attractive for *expensive* simulations; i.e., simulations that require relatively much computer time per scenario. Saturated resolution V designs were developed by Rechtschaffner (1967).

Central composite designs (CCD) are meant for the estimation of second-order polynomials. These designs augment resolution V designs with the base scenario and $2k$ scenarios that change factors one at a time; this changing increases and decreases each factor in turn. Saturated variants (smaller than CCD) are discussed in Kleijnen (1987, pp. 314–316).

The main conclusion is that *incomplete designs* for *low-order polynomial regression* are plentiful in both the classic DOE literature and the simulation literature. (The designs in the remainder of this chapter are more challenging.)

Designs for Simulations with Too Many Factors

3.4.4

Most practical, non-academic simulation models have many factors; for example, Kleijnen et al. (2003b) experiment with a supply-chain simulation model with nearly 100 factors. Even a Plackett–Burman design would then take 102 scenarios. Because each scenario needs to be replicated several times, the total computer time may then be prohibitive. For that reason, many analysts keep a lot of factors fixed (at their base values), and experiment with only a few remaining factors. An example is a military (agent-based) simulation that was run millions of times for just a few scenarios – changing only a few factors; see Horne and Leonardi (2001).

However, statisticians have developed designs that require fewer than k scenarios – called *supersaturated designs*; see Yamada and Lin (2002). Some designs *aggregate* the k individual factors into groups of factors. It may then happen that the effects of individual factors cancel out, so the analysts would erroneously conclude that all factors within that group are unimportant. The solution is to define the -1 and $+1$ levels of the individual factors such that all first-order effects β_j ($j = 1, \dots, k$) are *non-negative*. My experience is that in practice the users do know the direction of the first-order effects of individual factors.

There are several types of group screening designs; for a recent survey including references, I refer to Kleijnen et al. (2003b). Here I focus on the most efficient type, namely *Sequential Bifurcation* designs.

This design type is so efficient because it proceeds *sequentially*. It starts with only two scenarios, namely, one scenario with all individual factors at -1 , and a second scenario with all factors at $+1$. Comparing the outputs of these two extreme scenarios requires only two replications because the aggregated effect of the group factor is huge compared with the intrinsic noise (caused by the pseudorandom numbers). The next step splits – *bifurcates* – the factors into two groups. There are several heuristic rules to decide on how to assign factors to groups (again see Kleijnen et al. 2003b). Comparing the outputs of the third scenario with the outputs of the preceding scenarios enables the estimation of the aggregated effect of the individual factors within a group. Groups – and all its individual factors – are eliminated from further experimentation as soon as the group effect is statistically unimportant. Obviously, the groups get smaller as the analysts proceed sequentially. The analysts stop, once the first-order effects β_j of all the important individual factors are estimated. In their supply-chain simulation, Kleijnen et al. (2003b) classify only 11 of the 92 factors as important. (Next, this shortlist of important factors is further investigated to find a robust solution.)

3.5 Kriging

Let's return to the example in Fig. 3.1. If the analysts are interested in the input/output behavior within 'local area 1', then a first-order polynomial may be adequate. Maybe, a second-order polynomial is required to get a valid approximation in 'local area 2', which is larger and shows non-linear behavior of the input/output function. However, Kleijnen and Van Beers (2003a) present an example illustrating that the second-order polynomial gives very poor predictions – compared with Kriging.

Kriging has been often applied in deterministic simulation models. Such simulations are used for the development of airplanes, automobiles, computer chips, computer monitors, etc.; see Sacks et al. (1989)'s pioneering article, and – for an update – see Simpson et al. (2001). For Monte Carlo experiments, I do not know any applications yet. First, I explain the basics of Kriging; then DOE aspects.

3.5.1 Kriging Basics

Kriging is an *interpolation* method that predicts unknown values of a random process; see the classic textbook on Kriging in spatial statistics, Cressie (1993). More precisely, a Kriging prediction is a weighted linear combination of all output values already observed. These weights depend on the distances between the input for which the output is to be predicted and the inputs already simulated. Kriging assumes that *the closer the inputs are, the more positively correlated the outputs are*. This assumption is modeled through the correlogram or the related variogram, discussed below.

Note that in deterministic simulation, Kriging has an important advantage over regression analysis: Kriging is an *exact* interpolator; that is, predicted values at observed input values are exactly equal to the observed (simulated) output values. In random simulation, however, the observed output values are only estimates of the true values, so exact interpolation loses its intuitive appeal. Therefore regression uses OLS, which minimizes the residuals – squared and summed over all observations.

The simplest type of Kriging – to which I restrict myself in this chapter – assumes the following *metamodel* (also see (3.4) with $\mu = \beta_0$ and $\beta_1 = \beta_2 = \beta_3 = 0$):

$$y = \mu + e \quad \text{with} \quad (3.13a)$$

$$E(e) = 0, \quad \text{var}(e) = \sigma^2, \quad (3.13b)$$

where μ is the mean of the stochastic process $y(\cdot)$, and e is the additive noise, which is assumed to have zero mean and non-constant finite variance $\sigma^2(\mathbf{x})$ (furthermore, many authors assume normality). Kriging further assumes a *stationary covariance process*; i.e., μ and σ in (3.13a) are constants, and the covariances of $y(\mathbf{x} + \mathbf{h})$ and $y(\mathbf{x})$ depend only on the distance (or ‘lag’) between their inputs, namely $|\mathbf{h}| = |(\mathbf{x} + \mathbf{h}) - \mathbf{x}|$. (In deterministic simulation, the analysts assume that the *deterministic* input/output behavior can be adequately approximated by the *random* model given in (3.13b).)

The Kriging *predictor* for the unobserved input \mathbf{x}_0 – denoted by $\hat{y}(\mathbf{x}_0)$ – is a weighted linear combination of all the n output data already observed – $y(\mathbf{x}_i)$:

$$\hat{y}(\mathbf{x}_0) = \sum_{i=1}^n \lambda_i \cdot y(\mathbf{x}_i) = \boldsymbol{\lambda}' \cdot \mathbf{y} \quad \text{with} \quad (3.14a)$$

$$\sum_{i=1}^n \lambda_i = 1, \quad (3.14b)$$

where $\boldsymbol{\lambda} = (\lambda_1, \dots, \lambda_n)^T$ and $\mathbf{y} = (y_1, \dots, y_n)^T$.

To quantify the weights $\boldsymbol{\lambda}$ in (3.14), Kriging derives the *best linear unbiased estimator* (BLUE), which minimizes the Mean Squared Error (MSE) of the predictor:

$$\text{MSE}(\hat{y}(\mathbf{x}_0)) = E\left(\left(y(\mathbf{x}_0) - \hat{y}(\mathbf{x}_0)\right)^2\right)$$

with respect to $\boldsymbol{\lambda}$. Obviously, these weights depend on the covariances mentioned below (3.13). Cressie (1993) characterizes these covariances through the *variogram*, defined as $2\gamma(\mathbf{h}) = \text{var}(y(\mathbf{x} + \mathbf{h}) - y(\mathbf{x}))$. (I follow Cressie (1993), who uses variograms to express covariances, whereas Sacks et al. (1989) use correlation functions.) It can be proven that the *optimal* weights in (3.14) are

$$\boldsymbol{\lambda}^T = \left(\mathbf{y} + \mathbf{1} \frac{\mathbf{1} - \mathbf{1}^T \boldsymbol{\Gamma}^{-1} \mathbf{y}}{\mathbf{1}^T \boldsymbol{\Gamma}^{-1} \mathbf{1}} \right)^T \boldsymbol{\Gamma}^{-1} \quad (3.15)$$

with the following symbols:

$\boldsymbol{\gamma}$: vector of the n (co)variances between the output at the new input \boldsymbol{x}_0 and the outputs at the n old inputs, so $\boldsymbol{\gamma} = (\gamma(\boldsymbol{x}_0 - \boldsymbol{x}_1), \dots, \gamma(\boldsymbol{x}_0 - \boldsymbol{x}_n))^T$

$\boldsymbol{\Gamma}$: $n \times n$ matrix of the covariances between the outputs at the n old inputs – with element (i, j) equal to $\gamma(\boldsymbol{x}_i - \boldsymbol{x}_j)$

$\mathbf{1}$: vector of n ones.

I point out that the optimal weights defined by (3.15) vary with the input value for which output is to be predicted (see $\boldsymbol{\gamma}$), whereas linear regression uses the same estimated parameters $\hat{\boldsymbol{\beta}}$ for all inputs to be predicted.

3.5.2 Designs for Kriging

The most popular design type for Kriging is *Latin hypercube sampling* (LHS). This design type was invented by McKay, Beckman, and Conover (1979) for deterministic simulation models. Those authors did not analyze the input/output data by Kriging (but they did assume input/output functions more complicated than the low-order polynomials in classic DOE). Nevertheless, LHS is much applied in Kriging nowadays, because LHS is a simple technique (it is part of spreadsheet add-ons such as @Risk).

LHS offers *flexible* design sizes n (number of scenarios simulated) for any number of simulation inputs, k . A simplistic example is shown for $k = 2$ and $n = 4$ in Table 3.5 and Fig. 3.4, which are constructed as follows.

1. The table illustrates that LHS divides each input range into n intervals of equal length, numbered from 1 to n (in the example, we have $n = 4$; see the numbers in the last two columns); i.e., the number of values per input can be much larger than in the designs discussed in Sect. 3.4.
2. Next, LHS places these integers $1, \dots, n$ such that each integer appears exactly once in each row and each column of the design. (This explains the term ‘Latin hypercube’: it resembles Latin squares in classic DOE.)
3. Within each cell of the design in the table, the exact input value may be sampled uniformly; see Fig. 3.4. (Alternatively, these values may be placed systematically in the middle of each cell. In risk analysis, this uniform sampling may be replaced by sampling from some other distribution for the input values.)

Because LHS implies randomness, the resulting design may happen to include *outlier* scenarios (to be simulated). Furthermore, it might happen – with small probability – that in Fig. 3.4 all scenarios lie on the main diagonal, so the values of

Table 3.5. A LHS design for two factors and four scenarios

Scenario	Interval factor 1	Interval factor 2
1	2	1
2	1	4
3	4	3
4	3	2

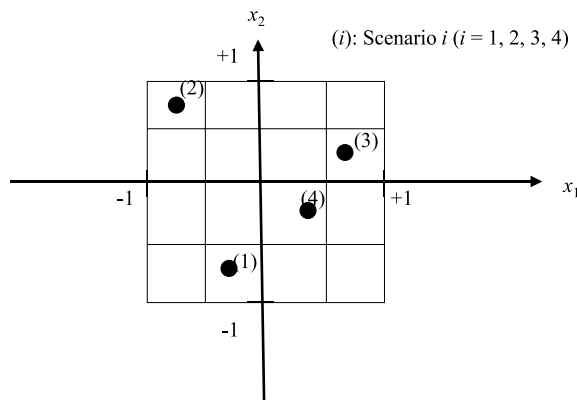


Figure 3.4. A LHS design for two factors and four scenarios

the two inputs have a correlation coefficient of -1 . Therefore LHS may be adjusted to give (nearly) orthogonal designs; see Ye (1998).

Let's compare classic designs and LHS geometrically. Figure 3.3 illustrates that many classic designs consists of corners of k -dimensional cubes. These designs imply simulation of *extreme scenarios*. LHS, however, has better *space filling* properties.

This property has inspired many statisticians to develop other space filling designs. One type maximizes the minimum Euclidean distance between any two points in the k -dimensional experimental area. Related designs minimize the maximum distance. See Koehler and Owen (1996), Santner et al. (2003), and also Kleijnen et al. (2003a).

Conclusions

3.6

Because simulation – treated as a black box – implies *experimentation* with a model, design of experiment is essential. In this chapter, I discussed both *classic* designs for low-order polynomial regression models and *modern* designs (including Latin hypercube sampling) for other metamodels such as Kriging models. The simpler the metamodel is, the fewer scenarios need to be simulated. (Cross validation of the metamodel selected, is discussed in Chap. III.1 by Wang.)

I did not discuss so-called *optimal designs* because these designs use statistical assumptions (such as white noise) that I find too unrealistic. A recent discussion of optimal designs including references is Spall (2003).

Neither did I discuss the designs in *Taguchi* (1987), as I think that the classic and modern designs (which I did discuss) are superior. Nevertheless, I think that Taguchi's concepts – as opposed to his statistical techniques – are important. In practice, the 'optimal' solution may break down because the environment turns out to differ from the environment that the analysts assumed when deriving the

optimum. Therefore they should look for a ‘robust’ solution. For further discussion I refer to Kleijnen et al. (2003a).

Because of space limitations, I did not discuss *sequential* designs, except for sequential bifurcation and two-stage resolution IV designs. Nevertheless, the sequential nature of simulation experiments (caused by the computer architecture) makes sequential designs very attractive. This is an area of active research nowadays; see Jin et al. (2002), Kleijnen et al. (2003a), and Kleijnen and Van Beers (2003b).

I mentioned several more research issues; for example, importance sampling. Another interesting question is: how much computer time should analysts spend on *replication*; how much on exploring *new* scenarios?

Another challenge is to develop designs that explicitly account for *multiple outputs*. This may be a challenge indeed in sequential bifurcation (depending on the output selected to guide the search, different paths lead to the individual factors identified as being important). In practice, multiple outputs are the rule in simulation; see Kleijnen et al. (2003a).

The application of *Kriging* to *random* simulation models (such models are a focus of this handbook, including this chapter) seems a challenge. Moreover, corresponding software needs to be developed. Current software focuses on deterministic simulation; see Lophaven et al. (2002).

Comparison of various metamodel types and their designs remains a major problem. For example, Meckesheimer et al. (2001) compare radial basis, neural net, and polynomial metamodels. Clarke et al. (2003) compare low-order polynomials, radial basis functions, Kriging, splines, and support vector regression. Alam et al. (2003) found that LHS gives the best neural-net metamodels. Comparison of screening designs has hardly begun; see Kleijnen et al. (2003 a,b).

References

- Alam, F.M., K.R. McNaught, T.J. Ringrose (2003). A comparison of experimental designs in the development of a neural network simulation metamodel. *Simulation Modelling: Practice and Theory*, accepted conditionally.
- Angün, E., D. den Hertog, G. Gürkan, J.P.C. Kleijnen (2002). Response surface methodology revisited. In: *Proceedings of the 2002 Winter Simulation Conference*, ed. E. Yücesan, C.H. Chen, J.L. Snowdon, J.M. Charnes, Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, 377–383.
- Antoniadis, A., D.T. Pham (1998). Wavelet regression for random or irregular design. *Computational Statistics and data Analysis* 28:353–369.
- Bettonvil, B., J.P.C. Kleijnen (1990). Measurement scales and resolution IV designs. *American Journal of Mathematical and Management Sciences* 10 (3–4): 309–322.
- Clarke, S.M., J.H. Griebisch, T.W., Simpson (2003). Analysis of support vector regression for approximation of complex engineering analyses. *Proceedings of DETC '03, ASME 2003 Design Engineering Technical Conferences and Computers and Information in Engineering Conference, Chicago*.

- Cressie, N.A.C (1993). *Statistics for spatial data*. New York: Wiley.
- Donohue, J. M., E.C. Houck, R.H. Myers (1993). Simulation designs and correlation induction for reducing second-order bias in first-order response surfaces *Operations Research* 41 (5):880–902.
- Efron, B. and R.J. Tibshirani (1993). *An introduction to the bootstrap*. New York: Chapman & Hall.
- Glasserman, P., P. Heidelberger, and P. Shahabuddin (2000), Variance reduction techniques for estimating value-at-risk. *Management Science*, 46, no. 10, pp. 1349–1364.
- Horne, G., M. Leonardi, eds (2001). *Maneuver warfare science 2001*. Quantico, Virginia: Defense Automatic Printing Service.
- Jin, R, W. Chen, and A. Sudjianto (2002). On sequential sampling for global metamodeling in engineering design. *Proceedings of D ETC '02, ASME 2002 Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, DETC2002/DAC-34092, September 29–October 2, 2002, Montreal, Canada.
- Kleijnen, J.P.C (1998). Experimental design for sensitivity analysis, optimization, and validation of simulation models. In: *Handbook of Simulation*, ed. J. Banks, 173–223. New York: Wiley.
- Kleijnen, J.P.C (1992). Regression metamodels for simulation with common random numbers: comparison of validation tests and confidence intervals. *Management Science* 38 (8): 1164–1185.
- Kleijnen, J.P.C (1987). *Statistical tools for simulation practitioners*. New York: Marcel Dekker.
- Kleijnen, J.P.C (1975). *Statistical techniques in simulation, volume II*. New York: Marcel Dekker. (Russian translation, Publishing House “Statistics”, Moscow, 1978).
- Kleijnen, J.P.C. S.M. Sanchez, T.W. Lucas, T.M. Cioppa (2003a). A user’s guide to the brave new world of designing simulation experiments. *INFORMS Journal on Computing* (accepted conditionally).
- Kleijnen, J.P.C., B. Bettonvil, F. Person (2003b). Finding the important factors in large discrete-event simulation: sequential bifurcation and its applications. In: *Screening*, ed. A.M. Dean, S.M. Lewis, New York: Springer-Verlag (forthcoming; preprint: <http://center.kub.nl/staff/kleijnen/papers.html>).
- Kleijnen, J.P.C. and R.Y. Rubinstein (2001). Monte Carlo sampling and variance reduction techniques. *Encyclopedia of Operations Research and Management Science*, Second edition, edited by S. Gass and C. Harris, Kluwer Academic Publishers, Boston, 2001, pp. 524–526.
- Kleijnen, J.P.C., R.G. Sargent (2000). A methodology for the fitting and validation of metamodels in simulation. *European Journal of Operational Research* 120 (1): 14–29.
- Kleijnen, J.P.C., W.C.M. Van Beers 2003a. Robustness of Kriging when interpolating in random simulation with heterogeneous variances: some experiments. *European Journal of Operational Research* (in press).

- Kleijnen, J.P.C., W.C.M. Van Beers. 2003b. Application-driven sequential designs for simulation experiments: Kriging metamodeling. *Journal Operational Research Society* (in press); preprint: <http://center.kub.nl/staff/kleijnen/papers.html>.
- Koehler, J.R., A.B. Owen (1996). Computer experiments. In: *Handbook of Statistics, Volume 13*, Eds. S. Ghosh, C.R. Rao, 261–308. Amsterdam: Elsevier.
- Law, A.M., W.D. Kelton (2000). *Simulation modeling and analysis*. 3rd ed. New York: McGraw-Hill
- Lophaven, S.N., H.B. Nielsen, and J. Sondergaard (2002). *DACE: a Matlab Kriging toolbox, version 2.0*. IMM Technical University of Denmark, Lyngby.
- McKay, M.D., R.J. Beckman, W.J. Conover (1979). A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics* 21 (2): 239–245 (reprinted in 2000: *Technometrics* 42 (1): 55–61).
- Meckesheimer, M., R.R. Barton, F. Limayem, B. Yannou (2001). Metamodeling of combined discrete/continuous responses. *AIAA Journal* 39 1950–1959.
- Rechtschaffner, R.L (1967). Saturated fractions of 2^n and 3^n factorial designs. *Technometrics* 9 569–575.
- Sacks, J., W.J. Welch, T.J. Mitchell, H.P. Wynn (1989). Design and analysis of computer experiments. *Statistical Science* 4 (4) 409–435.
- Santner, T.J., B.J. Williams, and W.I. Notz (2003), *The design and analysis of computer experiments*. New York: Springer-Verlag.
- Simpson, T.W., T.M. Mauery, J.J. Korte, F. Mistree (2001). Kriging metamodels for global approximation in simulation-based multidisciplinary design optimization. *AIAA Journal* 39 (12) 2233–2241.
- Spall, J.C (2003). *Introduction to stochastic search and optimization; estimation, simulation, and control*. Hoboken, New Jersey, Wiley.
- Taguchi, G. 1987. *System of experimental designs, Volumes 1 and 2*. White Plains, NY: UNIPUB/Krauss International.
- Yamada, S., Lin, D. K. J. (2002). Construction of mixed-level supersaturated design, *Metrika* (56), 205–214.
- Ye, K.Q. (1998). Orthogonal column Latin hypercubes and their application in computer experiments. *Journal Association Statistical Analysis, Theory and Methods*, (93) 1430–1439.
- Zeigler B.P., K. Praehofer, T.G. Kim (2000). *Theory of modeling and simulation. 2nd ed.* New York: Academic Press.

Multivariate Density Estimation and Visualization

III.4

David W. Scott

4.1	<i>Introduction</i>	518
4.2	<i>Visualization</i>	519
	Data Visualization	519
4.3	<i>Density Estimation Algorithms and Theory</i>	523
	A High-Level View of Density Theory.....	523
	The Theory of Histograms	526
	ASH and Kernel Estimators	528
	Kernel and Other Estimators.....	530
4.4	<i>Visualization of Trivariate Functionals</i>	532
4.5	<i>Conclusions</i>	534

Introduction

This chapter examines the use of flexible methods to approximate an unknown density function, and techniques appropriate for visualization of densities in up to four dimensions. The statistical analysis of data is a multilayered endeavor. Data must be carefully examined and cleaned to avoid spurious findings. A preliminary examination of data by graphical means is useful for this purpose. Graphical exploration of data was popularized by Tukey (1977) in his book on exploratory data analysis (EDA). Modern data mining packages also include an array of graphical tools such as the histogram, which is the simplest example of a density estimator. Exploring data is particularly challenging when the sample size is massive or if the number of variables exceeds a handful. In either situation, the use of nonparametric density estimation can aid in the fundamental goal of understanding the important features hidden in the data. In the following sections, the algorithms and theory of nonparametric density estimation will be described, as well as descriptions of the visualization of multivariate data and density estimates. For simplicity, the discussion will assume the data and functions are continuous. Extensions to discrete and mixed data are straightforward.

Statistical modeling of data has two general purposes: (1) understanding the shape and features of data through the density function, $f(\mathbf{x})$, and (2) prediction of y through the joint density function, $f(\mathbf{x}, y)$. When the experimental setting is well-known, parametric models may be formulated. For example, if the data are multivariate normal, $N(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, then the features of the density may be extracted from the maximum likelihood estimates of the parameters $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$. In particular, such data have one feature, which is a single mode located at $\boldsymbol{\mu}$. The shape of the data cloud is elliptical, and the eigenvalues and eigenvectors of the covariance matrix, $\boldsymbol{\Sigma}$, indicate the orientation of the data and the spread in those directions. If the experimental setting is not well-known, or if the data do not appear to follow a parsimonious parametric form, then nonparametric density estimation is indicated. The major features of the density may be found by counting and locating the sample modes. The shape of the density cannot easily be determined algebraically, but visualization methodology can assist in this task. Similar remarks apply in the regression setting.

When should parametric methods be used and when should nonparametric methods be used? A parametric model enjoys the advantages of well-known properties and parameters which may be interpreted. However, using parametric methods to explore data makes little sense. The features and shape of a normal fit will always be the same no matter how far from normal the data may be. Nonparametric approaches can fit an almost limitless number of density functional forms. However, at the model, parametric methods are always more statistically accurate than the corresponding nonparametric estimates. This statement can be made more precise by noting that parametric estimators tend to have lower variance, but are susceptible to substantial bias when the wrong parametric form is invoked. Nonparametric methods are not unbiased, but the bias asymptotically vanishes for

any continuous target function. Nonparametric algorithms generally have greater variance than a parametric algorithm. Construction of optimal nonparametric estimates requires a data-based approach in order to balance the variance and the bias, and the resulting mean squared error generally converges at a rate slower than the parametric rate of $O(n^{-1})$. In summary, nonparametric approaches are always appropriate for exploratory purposes, and should be used if the data do not follow a simple parametric form.

Visualization

4.2

Data Visualization

4.2.1

Visualization of data is a fundamental task in modern statistical practice. The most common figure for this purpose is the bivariate scatter diagram. Figure 4.1a displays the levels of blood fats in a sample of men with heart disease. The data have been transformed to a logarithm base 10 scale to minimize the effects of skewness. At a first glance, the data appear to follow a bivariate normal distribution. The sample correlation is only 0.22. One might examine each of the two variables separately as a univariate scatter diagram, which is commonly referred to as a “dot plot”, but such figures are rarely presented. Tukey advocated the histogram-like stem-and-leaf plot or the box-and-whiskers plot, which displays simple summaries including the median and quartiles. Figure 4.1b displays box-and-whisker plots for these variables. Clearly triglyceride values vary more than cholesterol and may still be right-skewed.

As shown later in Sect. 4.3.3, there may be rather subtle clusters within these data. The eye can readily detect clusters which are well-separated, but the eye is not reliable when the clusters are not well-separated, nor when the sample size is so large that the scatter diagram is too crowded. For example, consider the Old Faithful Geyser data (Azzalini and Bowman, 1990), (x_t, y_t) , where x_t measures the waiting time between successive eruptions of the geyser, and y_t measures the duration of the subsequent eruption. The data were blurred by adding uniform

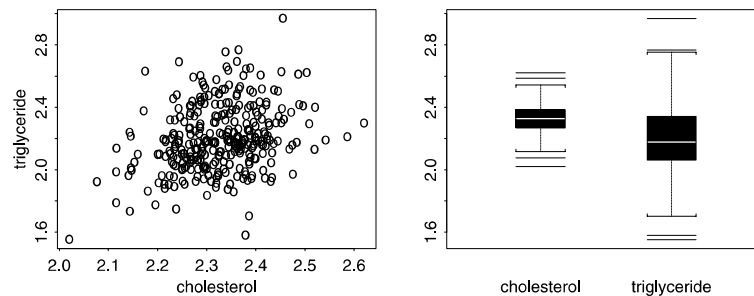


Figure 4.1. Cholesterol and triglyceride blood levels for 320 males with heart disease

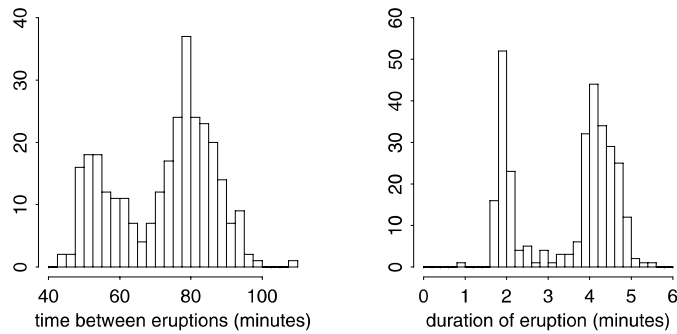


Figure 4.2. Waiting time and duration of 299 consecutive eruptions of the Old Faithful Geyser

noise to the nearest minute for x_t and to the nearest second for y_t . Figure 4.2 displays histograms of these two variables. Interestingly, neither appears to follow the normal distribution. The common feature of interest is the appearance of two modes. One group of eruptions is only 2 minutes in duration, while the other averages over 4 minutes in duration. Likewise, the waiting time between eruptions clusters into two groups, one less than an hour and the other greater than one hour. The distribution of eruption durations appears to be a mixture of two normal densities, but the distribution of the waiting times appears more complicated.

Finally, in Fig. 4.3 we examine the scatter diagrams of both (x_t, y_t) as well as the lagged values of eruption duration, (y_{t-1}, y_t) . The common feature in these two densities is the presence of three modes. As mentioned earlier, the eye is well-suited to discerning clusters that are well-separated. From Fig. 4.3a, short waiting periods are associated with long eruption durations. From Fig. 4.3b, all eruptions of short duration are followed by eruptions of long duration. Missing from Fig. 4.3b are any examples of eruptions of short duration following eruptions of short duration, which should be a plus for the disappointed tourist. The observant reader may notice an odd clustering of points at integer values of the eruption duration. A quick count shows that 23, 2, and 53 of the original 299 values occur exactly at $y = 2, 3,$ and 4 minutes, respectively. Examining the original time sequence suggests

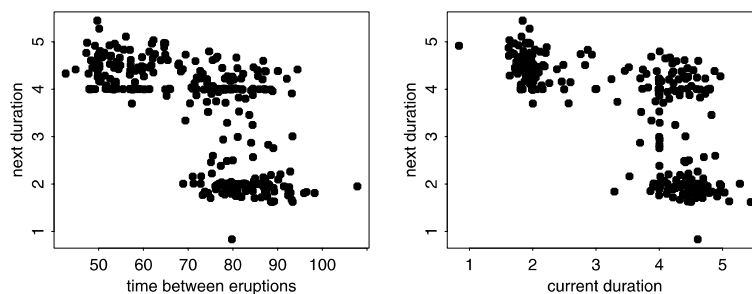


Figure 4.3. Two scatter diagrams of the Old Faithful Geyser data

that these measurements occur in clumps; perhaps accurate measurements were not taken after dark. Exploration of these data has revealed not only interesting features but also suggest possible data collection anomalies.

Massive datasets present different challenges. For example, the Landsat IV remote sensing dataset described by Scott (1992) contains information on 22,932 pixels of a scene imaged in 1977 from North Dakota. The variables displayed in Fig. 4.4 are the time of peak greenness of the crop in each pixel and the derived value of the maximum greenness, scaled to values 0–255 and blurred with uniform noise. Overplotting is apparent. Each successive figure drills down into the boxed region shown. Only 5.6% of the points are eliminated going to the second frame; 35.5% eliminated between the second and third frames; and 38.1% between the third and final frames, still leaving 8624 points. Overplotting is still apparent in the final frame. Generally, gleaning detailed density information from scatter diagrams is difficult at best. Nonparametric density estimation solves this problem.

To see the difficulty of gleaning density information from the graphs in Fig. 4.4, compare the bivariate histogram displayed in Fig. 4.5 for the data in frame (b) from Fig. 4.4. Using only the scatter diagram, there is no way to know the relative frequency of data in the two largest clusters except through the histogram.

The bivariate histogram uses rectangular-shaped bins. An interesting hybrid solution is to use hexagonal-shaped bins and to use a glyph to represent the bin count. Scott (1988) compared the statistical power of using squares, hexagons, and equilateral triangles as shapes for bins of bivariate histograms and concluded that hexagons were the best choice. Carr et al. (1992) examined the use of drawing a glyph in each bivariate bin rather than the perspective view. For graphical reasons,

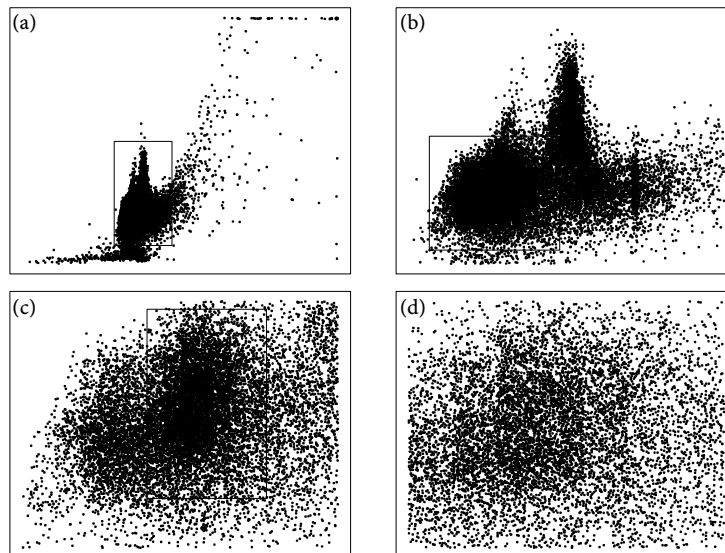


Figure 4.4. Drilling into the Landsat IV data with $n = 22932$

Carr found hexagonal bins were more effective. The bin count is represented by a hexagonal glyph whose area is proportional to the bin count. Figure 4.6 displays the hexagonal mosaic map of the same data as in Fig. 4.5. This representation gives a quite accurate summary of the density information. No bin counts are obscured as in the perspective view of the bivariate histogram.

In the next section, some of the algorithms for nonparametric density estimation and their theoretical properties are discussed. We then return to the visualization of data in higher dimensions.

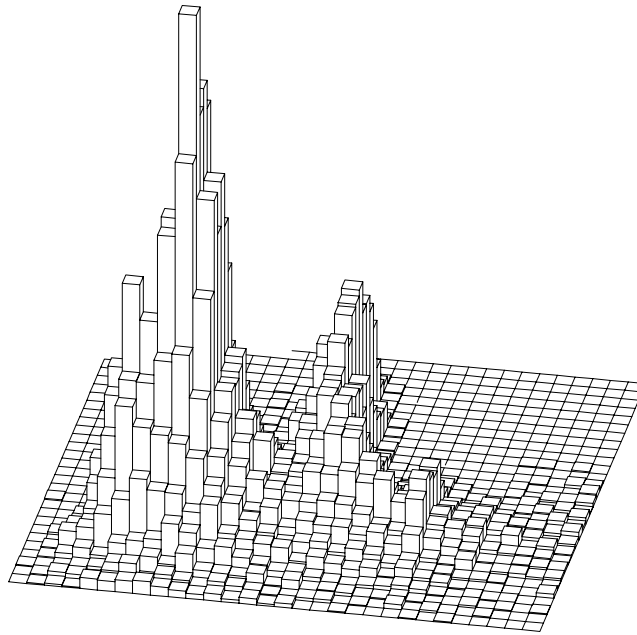


Figure 4.5. Histogram of data in Fig. 4.4b

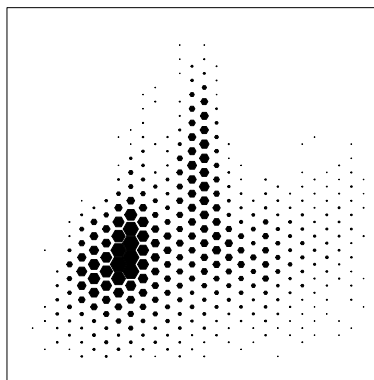


Figure 4.6. Hexagonal bin glyph of the data in Fig. 4.4b

Density Estimation Algorithms and Theory

4.3

This section includes enough algorithms and results to obtain a basic understanding of the opportunities and issues. Fortunately, there have been a number of readable monographs available for the reader interested in pursuing this subject in depth. In rough chronological order, excluding books primarily dealing with nonparametric regression, the list includes Tapia and Thompson (1978), Wertz (1978), Prakasa Rao (1983), Devroye and Györfi (1985), Silverman (1986), Devroye (1987), Nadaraya (1989), Härdle (1990), Scott (1992), Tarter and Lock (1993), Wand and Jones (1995), Simonoff (1996), Bowman and Azzalini (1997), and Tarter (2000).

The purpose of the next section is to provide a survey of important results without delving into the theoretical underpinnings and details. The references cited above are well-suited for that purpose.

A High-Level View of Density Theory

4.3.1

Smoothing Parameters

Every algorithm for nonparametric density estimation has one or more design parameters which are called the smoothing parameter(s) or bandwidth(s) of the procedure. The smoothing parameter controls the final appearance of the estimate. For an equally-spaced histogram, the bin width plays the primary role of a smoothing parameter. Of course, the bins may be shifted and the location of the bin edges is controlled by the bin origin, which plays the role of a secondary smoothing parameter. For a kernel estimator, the scale or width of the kernel serves as the smoothing parameter. For an orthogonal series estimator, the number of basis functions serves as the smoothing parameter. The smoothing parameters of a spline estimator also include the location of the knots. Similarly, a histogram with completely flexible bin widths has many more than two smoothing parameters.

No Unbiased Density Estimators

As a point estimator of $f(x)$, Rosenblatt (1956) proved that every nonparametric density estimator, $\hat{f}(x)$ is biased. However, it is usually true that the integral of all of the pointwise biases is 0. Thus mean squared error (MSE) and integrated mean squared error (IMSE) are the appropriate criteria to optimize the tradeoff between pointwise/integrated variance and squared bias.

Nonparametric density estimators always underestimate peaks and overestimate valleys in the true density function. Intuitively, the bias is driven by the degree of curvature in the true density. However, since the bias function is continuous and integrates to 0, there must be a few points where the bias does vanish. In fact, letting the smoothing parameter vary pointwise, there are entire intervals where the bias vanishes, including the difficult-to-estimate tail region. This

fact has been studied by Hazelton (1996) and Sain and Scott (2002). Since the bias of a kernel estimator does not depend on the sample size, these zero-bias or bias-annihilating estimates have more than a theoretical interest. However, there is much more work required for practical application. Alternatively, in higher dimensions away from peaks and valleys, one can annihilate pointwise bias by balancing directions of positive curvature against directions of negative curvature; see Terrell and Scott (1992). An even more intriguing idea literally adjusts the raw data points towards peaks and away from valleys to reduce bias; see Choi and Hall (1999).

Rates of Convergence

The rate of convergence of a nonparametric density estimator to the true density is much slower than in the parametric setting, assuming in the latter case that the correct parametric model is known. If the correct parametric model is not known, then the parametric estimates will converge but the bias will not vanish. The convergence is slower still in high dimensions, a fact which is often referred to as the *curse of dimensionality*. Estimating the derivative of a density function is even harder than coping with an additional dimension of data.

If the k -th derivative of a density is known to be smooth, then it is theoretically possible to construct an order- k nonparametric density estimation algorithm. The pointwise bias is driven by the k -th derivative at x , $f^{(k)}(x)$. However, if $k > 2$, then the density estimate will take on negative values for some points, x . It is possible to define higher-order algorithms which are non-negative, but these estimates do not integrate to 1; see Terrell and Scott (1980). Thus higher-order density estimation algorithms violate one of the two conditions for a true density: $f(x) \geq 0$ and $\int_{-\infty}^{\infty} f(x) dx = 1$. Of course, there are cases where the first condition is violated for lower-order estimators. Two such cases include orthogonal series estimators (Kronmal and Tarter, 1968; Watson, 1969) and boundary-corrected kernel estimators (Rice, 1984). Note that positive kernel estimators correspond to $k = 2$. Wahba (1981) studied the efficacy of higher-order procedures and suggested $k = 3$ often provided superior estimates. Scott (1992) also studied this question and found some improvement when $k = 3$, which must be traded off against the disadvantages of negative estimates.

Choosing Bandwidths in Practice

Picking the best smoothing parameter from data is an important task in practice. If the smoothing parameter is too small, the estimate is too noisy, exhibiting high various and extraneous wiggles. If the smoothing parameter is too large, then the estimate may miss key features due to oversmoothing, washing out small details. Such estimates have low variance but high bias in many regions. In practice, bandwidths that do not differ by more than 10–15% from the optimal bandwidth are usually satisfactory.

A statistician experienced in EDA is likely to find all estimates informative for bandwidths ranging from undersmoothed to oversmoothed. With a complicated

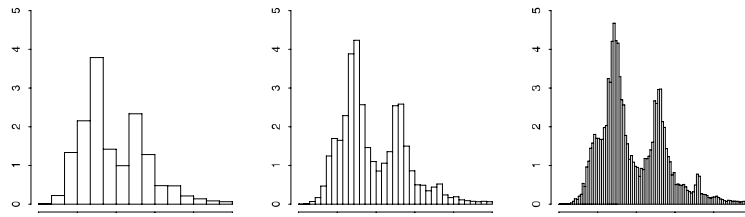


Figure 4.7. Histograms of x variable in Fig. 4.4b with 15, 35, and 100 bins

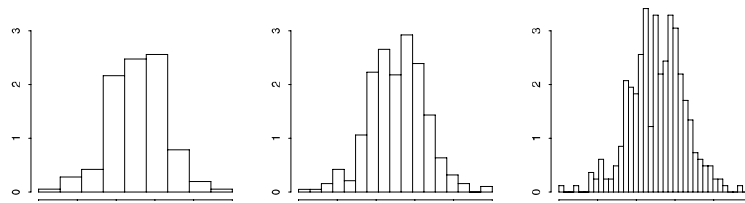


Figure 4.8. Histograms of \log_{10} -cholesterol variable in Fig. 4.1 with 9, 19, and 39 bins

density function, no single choice for the bandwidth may properly represent the density for all values of x . Thus the same bandwidth may result in undersmoothing for some intervals of x , oversmoothing in another interval, and yet near optimal smoothing elsewhere. However, the practical difficulty of constructing locally adaptive estimators makes the single-bandwidth case of most importance. Simple transformations of the data scale can often be an effective strategy (Wand et al., 1991). This strategy was used with the lipid data in Fig. 4.1, which were transformed to a \log_{10} scale.

Consider the 21,640 x points shown in frame (b) of Fig. 4.4. Histograms of these data with various numbers of bins are shown in Fig. 4.7. With so much data, the oversmoothed histogram Fig. 4.7a captures the major features, but seems biased downwards at the peaks. The final frame shows a histogram that is more useful for finding data anomalies than as a good density estimate.

The differences are more apparent with a smaller sample size. Consider the 320 \log_{10} -cholesterol levels shown in Fig. 4.1. Three histograms are shown in Fig. 4.8. The extra one or two modes are at least suggested in the middle panel, while the histogram in the first panel only suggests a rather unusual non-normal appearance. The third panel has many large spurious peaks. We conclude from these two figures that while an oversmoothed estimator may have a large error relative to the optimal estimator, the absolute error may still be reasonably small for very large data samples.

Oversmoothed Bandwidths

While there is no limit on how complicated a density may be (for which $\int f^{(k)}(x)^2 dx$ may grow without bound), the converse is not true. Terrell and Scott (1985) and Terrell (1990) show that for a particular scale of a density (for example, the range,

standard deviation, or interquartile range), there is in fact a lower bound among continuous densities for the roughness quantity

$$R(f^{(k)}) = \int_{-\infty}^{\infty} f^{(k)}(x)^2 dx. \quad (4.1)$$

In a real sense, such densities are the smoothest possible and are the easiest to estimate. The optimal bandwidth for these “oversmoothed densities” serves as an upper bound. Specifically, any other density with the same scale will have more complicated structure and will require a smaller bandwidth to more accurately estimate those features. Since oversmoothed bandwidths (and reference bandwidths as well) only use the data to estimate the scale (variance, for example), these data-based estimates are quite stable. Obtaining similar highly stable data-based nearly optimal bandwidth estimators requires very sophisticated estimates of the roughness function given in 4.1. One algorithm by Hall et al. (1991) is often highly rated in practice (Jones et al., 1996). seemed closely related to the oversmoothed bandwidths. These approaches all rely on asymptotic expansions of IMSE rather than an unbiased risk estimate, which underlies the least-squares or unbiased cross-validation algorithm introduced by Rude-mo (1982) and Bowman (1984). However, the unbiased risk approach has numerous extensions; see Sain and Scott (1996) and Scott (2001). Another algorithm that should be mentioned is the bootstrap bandwidth. For a Gaussian kernel, the bootstrap with an infinite number of repetitions has a closed form expression; see Taylor (1989). Multivariate extensions are discussed by Sain et al. (1994).

Many details of these ideas may be found in the literature and in the many textbooks available. The following section provides some indication of this research.

4.3.2 The Theory of Histograms

The basic results of density estimation are perhaps most easily understood with the ordinary histogram. Thus more time will be spent on the histogram with only an outline of results for more sophisticated and more modern algorithms.

Given an equally spaced mesh $\{t_k\}$ over the entire real line with $t_{k+1} - t_k = h$, the density histogram is given by

$$\hat{f}(x) = \frac{v_k}{nh} \quad \text{for } t_k < x < t_{k+1}, \quad (4.2)$$

where v_k is the number of data points falling in the k -th bin. Clearly, $\sum_k v_k = n$ and v_k is a Binomial random variable with mean $p_k = \int_{t_k}^{t_{k+1}} f(x) dx$; hence, $E v_k = n p_k$ and $\text{Var} v_k = n p_k (1 - p_k)$. Thus the pointwise variance of the histogram (4.2) is $n p_k (1 - p_k) / (n h)^2$, which is constant for all x in the k -th bin. Thus, the integrated variance (IV) over $(-\infty, \infty)$ may be found by integrating the pointwise variance over the k -th bin (i.e., multiply by the bin width h), and summing over all bins:

$$IV = \sum_{k=-\infty}^{\infty} \frac{np_k(1-p_k)}{(nh)^2} \times h = \sum_{k=-\infty}^{\infty} \frac{p_k(1-p_k)}{nh} = \frac{1}{nh} - \sum_k \frac{pk^2}{nh}, \quad (4.3)$$

since $\sum p_k = \int_{-\infty}^{\infty} f(x) dx = 1$. The final term may be shown to approximate $n^{-1} \int f(x)^2 dx$, which is asymptotically negligible. Thus the global integrated variance of the histogram can be controlled by collecting more data or choosing a wider bin width.

Next consider the bias of \hat{f} at a fixed point, x , which is located in the k -th bin. Note that $E\hat{f}(x) = np_k/nh = p_k/h$. A useful approximation to the bin probability is

$$p_k = \int_{t_k}^{t_{k+1}} f(y) dy = hf(x) + h^2 \left(\frac{1}{2} - \frac{x-t_k}{h} \right) f'(x) + \dots, \quad (4.4)$$

replacing the integrand $f(y)$ by $f(x) + (y-x)f'(x) + \dots$. Thus the pointwise bias may be approximated by

$$\text{Bias}\hat{f}(x) = E\hat{f}(x) - f(x) = \frac{p_k}{h} - f(x) = h \left(\frac{1}{2} - \frac{x-t_k}{h} \right) f'(x) + \dots. \quad (4.5)$$

Therefore, the bias is controlled by the first derivative of the unknown density at x . Since $t_k < x < t_{k+1}$, then the factor $(1/2 - (x - t_k)/h)$ in (4.5) varies from $-1/2$ to $1/2$. Thus the bias is also directly proportional to the bandwidth, h . To control the bias of the histogram estimate, the bandwidth h should be small. Comparing (4.3) and (4.5), the global consistency of the histogram can be guaranteed if, as $n \rightarrow \infty$, $h \rightarrow 0$ while ensuring that the product $nh \rightarrow \infty$ as well, for example, if $h = 1/\sqrt{n}$ (see Duda and Hart, 1973).

A more complete analysis of the bias (Scott, 1979) shows that the integrated squared bias is approximately $h^2 R(f')/12$, where $R(f') = \int f'(x)^2 dx$, so that the IMSE is given by

$$\text{IMSE} [\hat{f}_k] = \frac{1}{nh} + \frac{1}{12}h^2R(f') + O(n^{-1}). \quad (4.6)$$

From this equation, the optimal bandwidth is seen to be

$$h^* = \left[\frac{6}{nR(f')} \right]^{1/3} \quad \text{and} \quad \text{IMSE}^* = \left(\frac{9}{16} \right)^{1/3} R(f')^{1/3} n^{-2/3}. \quad (4.7)$$

Thus the optimal bandwidth approaches zero at the rate $O(n^{-1/3})$ and not the rate $O(n^{-1/2})$ as suggested by Duda and Hart (1973) nor the rate $O(1/\log n)$ as suggested by Sturges (1926). With regards to IMSE, the best rate a histogram can achieve is of order $O(n^{-2/3})$, which falls well short of the parametric rate of $O(n^{-1})$. From (4.7), the larger the value of the roughness $R(f')$ of the true density, the smaller the optimal bandwidth and the larger the average error.

Finally, the smoothest density with variance σ^2 is

$$g(x) = \frac{15}{16\sqrt{7}\sigma} \left(1 - \frac{x^2}{7\sigma^2} \right)^2 \quad -\sqrt{7}\sigma < x < \sqrt{7}\sigma \quad (4.8)$$

and zero elsewhere, for which $R(g') = 15\sqrt{7}/(343\sigma^3)$. Since $R(f') \geq R(g')$ for any other continuous density, f ,

$$h^* = \left[\frac{6}{nR(f')} \right]^{1/3} \leq \left[\frac{6}{nR(g')} \right]^{1/3} = \left[\frac{686\sigma^3}{5\sqrt{7}n} \right]^{1/3} = 3.73\sigma n^{-1/3}, \quad (4.9)$$

which is the “oversmoothed bandwidth” rule. Consider the normal reference rule, $f = \phi = N(\mu, \sigma^2)$, for which $R(\phi') = 1/(4\sqrt{\pi}\sigma^3)$, which when substituted into (4.7) gives $h^* = 3.49\sigma n^{-1/3}$, a value that is only 6.4% narrower than the oversmoothed bandwidth.

The oversmoothing rule (4.9) may be inverted when the scale is the range of the density to obtain a lower bound of $\sqrt[3]{2n}$ on the number of bins in the optimal histogram. This formula should be compared to Sturges’ rule of $1 + \log_2 n$, which is in common use in many statistical packages (Sturges, 1926). In fact, the histograms in the first frames of Figs. 4.7 and 4.8 correspond to Sturges’ rule, while the second frames of these figures correspond to the oversmoothed bandwidths. Presumably the optimal bandwidth would occur somewhere between the second and third frames of these figures. Clearly Sturges’ rule results in oversmoothed graphs since the optimal number of bins is severely underestimated.

4.3.3 ASH and Kernel Estimators

The histogram is an order-one density estimator, since the bias is determined by the first derivative. The estimators used most in practice are order-two estimators. (Recall that order-three estimators are not non-negative.) Perhaps the most unexpected member of the order-two class is the frequency polygon (FP), which is the piecewise linear interpolant of the midpoints of a histogram. (Scott, 1985a) showed that

$$\text{IMSE} \left[\widehat{f}_{\text{FP}} \right] = \frac{2}{3nh} + \frac{49}{2880} h^4 R(f'') + O(n^{-1}). \quad (4.10)$$

Compared to Equation (4.6), the integrated variance of a FP is 33% smaller and the integrated squared bias is two orders of magnitude smaller. Clearly, $h^* = O(n^{-1/5})$ and $\text{IMSE}^* = O(n^{-4/5})$. Thus the error converges at a faster rate than the histogram, by using bins which are wider and an estimator which is not discontinuous. Examples and other results such as oversmoothing may be found in Scott (1992).

The use of wider bins means that the choice of the bin origin has a larger impact, at least subjectively. Given a set of m shifted histogram, $\widehat{f}_1(x), \dots, \widehat{f}_m(x)$, one might use cross-validation to try to pick the best one. Alternatively, Scott (1985b) suggested the averaged shifted histogram (ASH), which is literally defined:

$$\widehat{f}_{\text{ASH}}(x) = \frac{1}{m} \sum_{k=1}^m \widehat{f}_k(x). \quad (4.11)$$

To be specific, suppose the collection of m histograms has meshes shifted by an amount $\delta = h/m$ from each other. Recompute the bin counts, v_k , on the finer mesh, $t'_k = k\delta$, $-\infty < k < \infty$. Then a bin count for one of the histograms with bin width h may be computed by adding m of the bin counts on the finer mesh. For x in the ℓ -th (narrow) bin, there are m shifted histograms that include the (narrow) bin count, v_ℓ . Adding these m shifted histograms together and averaging gives:

$$\frac{v_{\ell+1-m} + 2v_{\ell+2-m} + \dots + m v_\ell + \dots + 2v_{\ell+m-2} + v_{\ell+m-1}}{m \times nh}, \tag{4.12}$$

or after re-arranging

$$\hat{f}_{\text{ASH}}(x) = \frac{1}{nh} \sum_{k=1-m}^{m-1} \left(1 - \frac{|k|}{m}\right) v_{\ell+k}. \tag{4.13}$$

As the number of shifted histograms $m \rightarrow \infty$, the weights on the bin counts approaches the triangular kernel given by $K(t) = 1 - |t|$ for $|t| < 1$ and zero elsewhere. The ASH may be generalized to handle general weights by sampling from an arbitrary kernel function, $K(t)$, which is any symmetric probability density defined on the interval $[-1, 1]$. In this case,

$$\hat{f}_{\text{ASH}}(x) = \frac{1}{nh} \sum_{k=1-m}^{m-1} w_m(k) v_{\ell+k} \quad \text{where } w_m(k) \propto K(k/m). \tag{4.14}$$

Like the FP, the ASH is an order-two algorithm, but more efficient in the statistical sense.

In Fig. 4.9, two ASHs of the \log_{10} -cholesterol data are shown. The bin edge effects and discontinuities apparent in the ordinary histogram in Fig. 4.8 are removed. The extra features in the distribution are hinted at.

The extension of the ASH to bivariate (and multivariate) data is straightforward. A number of bivariate (multivariate) histograms are constructed with equal shifts along the coordinate axes and then averaged together. Figure 4.10 displays a bivariate ASH of the same lipid data displayed in Fig. 4.1. The strong bimodal and weak trimodal features are evident. The third mode is perhaps more clearly represented in a perspective plot; see Fig. 4.11. (Note that for convenience, the data were rescaled to the intervals $(0, 1)$ for these plots, unlike Fig. 4.1.) The precise

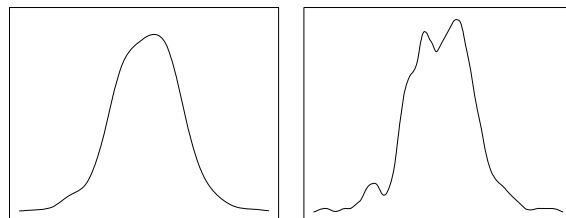


Figure 4.9. Averaged shifted histograms of the \log_{10} -cholesterol data

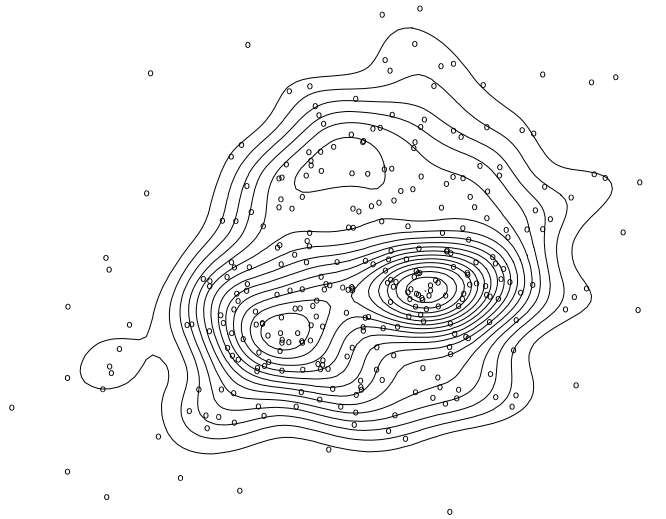


Figure 4.10. Bivariate ASH of the \log_{10} -cholesterol and \log_{10} -triglyceride data

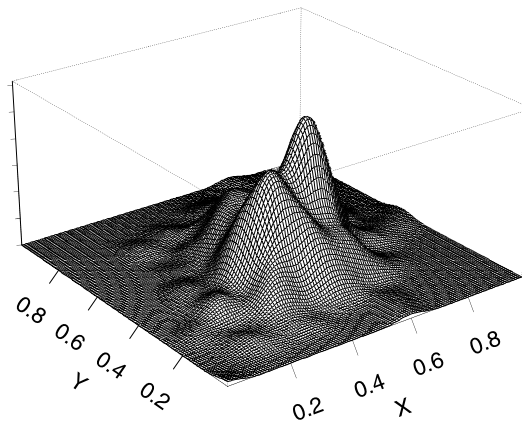


Figure 4.11. Perspective view of the Bivariate ASH in Fig. 4.10

location of the third mode above (and between) the two primary modes results in the masking of the multiple modes when viewed along the cholesterol axis alone. This masking feature is commonplace and a primary reason for trying to extend the dimensions available for visualization of the density function.

4.3.4 Kernel and Other Estimators

The ASH is a discretized representation of a kernel estimator. Binned kernel estimators are of great interest to reduce the computational burden. An alternative to the ASH is the fast Fourier transform approach of Silverman (1982). Kernel methods were introduced by Rosenblatt (1956) and Parzen (1962) with earlier work by

Evelyn Fix and Joe Hodges completed by 1951 in San Antonio, Texas (see Silverman and Jones, 1989).

Given a kernel function, $K(t)$, which is generally taken to be a symmetric probability density function, the kernel density estimate is defined by

$$\hat{f}_K(x) = \frac{1}{nh} \sum_{i=1}^n K\left(\frac{x - x_i}{h}\right) = \frac{1}{n} \sum_{i=1}^n K_h(x - x_i), \quad (4.15)$$

letting K_h denote the kernel density transformed by the scale factor, h ; that is, $K_h(t) = K(t/h)/h$. Among kernels with finite support, Beta densities shifted to the interval $(-1, 1)$ are popular choices. Among kernels with infinite support, the normal density is by far the most common choice. An important paper by Silverman (1981) showed that the normal kernel has the unique property that the number of modes in the kernel estimate monotonically decreases as the smoothing parameter increases. For many exploratory purposes, this property alone is reason to use only the normal kernel. Minnotte and Scott (1993) proposed graphing the locations of all modes at all bandwidths in the “mode tree.” Minnotte (1997) proposed an extension of Silverman’s bootstrap test (Silverman, 1981) for the number of modes to test individual modes. Software for the ASH, kernel estimates, and the various mode tests may be found on the web; see statlib at www.stat.cmu.edu, for example.

Multivariate extensions of the kernel approach generally rely upon the product kernel. For example, with bivariate data $\{(x_i, y_i), i = 1, \dots, n\}$, the bivariate (product) kernel estimator is

$$\hat{f}_K(x, y) = \frac{1}{n} \sum_{i=1}^n K_{h_x}(x - x_i) K_{h_y}(y - y_i). \quad (4.16)$$

A different smoothing parameter for each variable generally gives sufficient control. A full bivariate normal kernel may be used in special circumstances, effectively adding one additional smoothing parameter in the form of the correlation coefficient. However, an equivalent estimate may be obtained by rotating the data so that the correlation in the kernel vanishes, so that the product kernel may be used on the transformed data.

In higher dimensions, some care must be exercised to minimize the effects of the curse of dimensionality. First, marginal variable transformations should be explored to avoid a heavily skewed appearance or heavy tails. Second, a principal components analysis should be performed to determine if the data are of full rank. If so, the data should be projected into an appropriate subspace. No nonparametric procedure works well if the data are not of full rank. Finally, if the data do not have many significant digits, the data should be carefully blurred. Otherwise the data may have many repeated values, and cross-validation algorithms may believe the data are discrete and suggest using $h = 0$. Next several kernel or ASH estimates may be calculated and explored to gain an understanding of the data, as a preliminary step towards further analyses.

An extensive body of work also exists for orthogonal series density estimators. Originally, the Fourier basis was studied, but more modern choices for the basis functions include wavelets. These can be re-expressed as kernel estimators, so we do not pursue these further. In fact, a number of workers have shown how almost any nonparametric density algorithm can be put into the form of a kernel estimator; see Walter and Blum (1979) and Terrell and Scott (1992), for example. More recent work on local likelihood algorithms for density estimation further shows how closely related parametric and nonparametric thinking really is; see Loader (1999) for details and literature.

4.4 Visualization of Trivariate Functionals

The field of scientific visualization has greatly enhanced the set of tools available for the statistician interested in exploring the features of a density estimate in more than two dimensions. In this section, we demonstrate by example the exploration of trivariate data.

We continue our analysis of the data given by the duration of 299 consecutive eruptions of the Old Faithful geyser. A graph of the histogram of these data is displayed in Fig. 4.2b. We further modified the data as follows: the 105 values that were only recorded to the nearest minute were blurred by adding uniform noise of 30 seconds in duration. (The remaining data points were recorded to the nearest second). An easy way to generate high-dimensional data from a univariate time series is to group adjacent values. In Fig. 4.12, ASH's of the univariate data $\{y_t\}$ and the lagged data $\{(y_{t-1}, y_t)\}$ are shown. The obvious question is whether knowledge of y_{t-1} is useful for predicting the value of y_t . Clearly, the answer is in the affirmative, but the structure would not be well-represented by an autoregressive model.

Next, we computed the ASH for the trivariate lagged data $\{(y_{t-2}, y_{t-1}, y_t)\}$. The resulting estimate, $\hat{f}_{\text{ASH}}(y_{t-2}, y_{t-1}, y_t)$, may be explored in several fashions. The question is whether knowing y_{t-2} can be used to predict the joint behavior of (y_{t-1}, y_t) . This may be accomplished, for example, by examining *slices* of the trivariate densi-

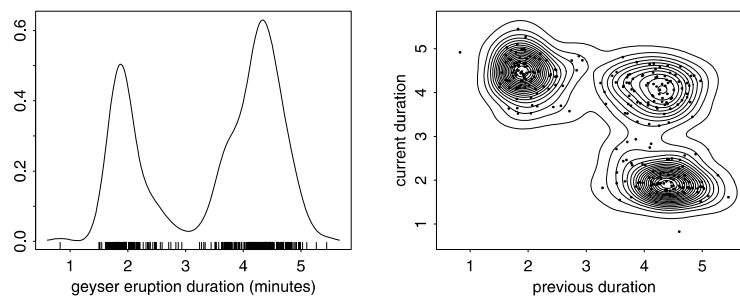


Figure 4.12. Averaged shifted histograms of the Old Faithful geyser duration data

ty. Since the (univariate) density has two modes at $x = 1.88$ and $x = 4.33$ minutes, we examine the slices $\hat{f}_{\text{ASH}}(1.88, y_{t-1}, y_t)$ and $\hat{f}_{\text{ASH}}(4.33, y_{t-1}, y_t)$; see Fig. 4.13. The 297 data points were divided into two groups, depending on whether $y_{t-2} < 3.0$ or not. The first group of points was added to Fig. 4.13a, while the second group was added to Fig. 4.13b.

Since each axis was divided into 100 bins, there are 98 other views one might examine like Fig. 4.13. (An animation is actually quite informative.) However, one may obtain a holistic view by examining level sets of the full trivariate density. A level set is the set of all points \mathbf{x} such that $\hat{f}_{\text{ASH}}(\mathbf{x}) = \alpha \hat{f}_{\text{max}}$, where \hat{f}_{max} is the maximum or modal value of the density estimate, and $\alpha \in (0, 1)$ is a constant that determines the contour level. Such contours are typically smooth surfaces in \mathbb{R}^3 . When $\alpha = 1$, then the “contour” is simply the modal location point. In Fig. 4.14, the contour corresponding to $\alpha = 58\%$ is displayed. Clearly these data are multimodal, as five well-separated high-density regions are apparent. Each cluster corresponds to a different sequence of eruption durations, such as long-long-long.

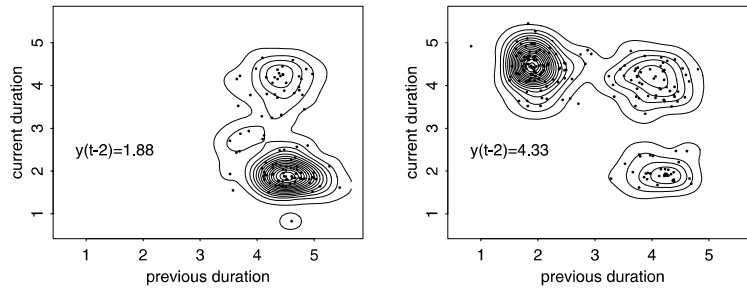


Figure 4.13. Slices of the trivariate averaged shifted histogram of lagged values of the Old Faithful geyser duration data

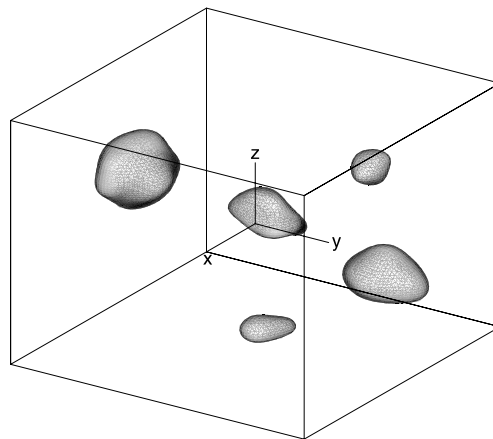


Figure 4.14. Visualization of the $\alpha = 58\%$ contour of the trivariate ASH of the lagged geyser duration data

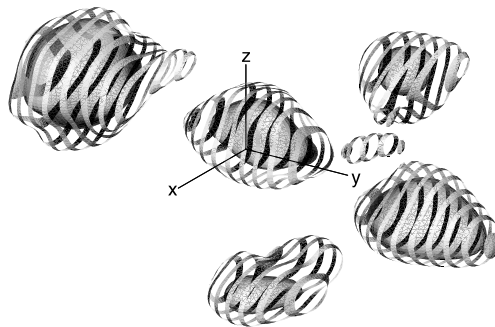


Figure 4.15. Visualization of the $\alpha = 28\%$ and 58% contours of the trivariate ASH of the lagged geyser duration data

The five clusters are now also quite apparent in both frames of Fig. 4.13. Of the eight possible sequences, three are not observed in this sequence of 299 eruptions.

A single contour does not convey as much information as several. Depending on the display device, one may reasonably view three to five contours, using transparency to see the higher density contours that are “inside” the lower density contours. Consider adding a second contour corresponding to $\alpha = 28\%$ to that in Fig. 4.14. Rather than attempt to use transparency, we choose an alternative representation which emphasizes the underlying algorithms. The software which produced these figures is called *ashn* and is available at the author’s website. ASH values are computed on a three-dimensional lattice. The surfaces are constructed using the marching cubes algorithm (Lorensen and Cline, 1987), which generates thousands of triangles that make up each surface. In Fig. 4.15, we choose not to plot all of the triangles but only every other “row” along the second axis. The striped effect allows one to interpolate and complete the low-density contour, while allowing one to look inside and see the high-density contour. Since there are five clusters, this is repeated five times. A smaller sixth cluster is suggested as well.

4.5 Conclusions

Exploring data is an important part of successful statistical model building. General discussions of graphical tools may be found in Tufté (1983), Wainer (1997), Cleveland (1985, 1993), Wegman and Depriest (1986) and Buja and Tukey (1991), for example. Advanced exploratory software may be found in many commercial packages, but of special note is the XGobi (Swayne et al., 1991) system and successors. Immersive environments are also of growing interest (Cook et al., 1997). A general visualization overview may be found in Wolff and Yaeger (1993).

Especially when the data size grows, point-oriented methods should be supplemented by indirect visualization techniques based upon nonparametric density estimation or by parallel coordinates (Inselberg, 1985; Wegman, 1990). Many den-

sity algorithms are available. The use of order-two algorithms is generally to be recommended. These should be calibrated by several techniques, starting with an oversmoothed bandwidth and the normal reference rule.

For data beyond three dimensions, density estimates may be computed and slices such as $\hat{f}(x, y, z, t = t_0)$ visualized. If advanced hardware is available, the surfaces can be animated as t varies continuously over an interval (t_0, t_1) ; see Scott (1986; 2000). Obviously, this is most useful for data in four and five dimensions. In any case, multivariate density estimation and visualization are important modern tools for EDA.

Acknowledgments. This research was supported in part by the National Science Foundation grants NSF EIA-9983459 (digital government) and DMS 02-04723 (non-parametric methodology).

References

- Azzalini, A. and Bowman, A.W. (1990). A Look at Some Data on the Old Faithful Geysers. *Applied Statistics*, 39:357–365.
- Bowman, A.W. (1984). An Alternative Method of Cross-Validation for the Smoothing of Density Estimates. *Biometrika*, 71:353–360.
- Bowman, A.W. and Azzalini, A. (1990). *Applied Smoothing Techniques For Data Analysis: The Kernel Approach With S-Plus Illustrations*, Oxford University Press.
- Buja, A. and Tukey, P.A., eds. (1991). *Computing and Graphics in Statistics*, Springer-Verlag Inc, New York.
- Carr, D.B., Olsen, A.R. and White, D. (1992). Hexagon Mosaic Maps for the Display of Univariate and Bivariate Geographical Data. *Cartograph. Geograph. Information Systems*, 19:228–231.
- Choi, E. and Hall, P. (1999). Data Sharpening as a Prelude to Density Estimation. *Biometrika*, 86:941–947.
- Cleveland, W.S. (1985). *The Elements of Graphing Data*, Wadsworth, Monterey, CA.
- Cleveland, W.S. (1993). *Visualizing Data*, Hobart Press, Summit, NJ.
- Cook, D., Cruz-Neira, C., Kohlmeyer, B.D., Lechner, U., Lewin, N., Elson, L., Olsen, A., Pierson, S. and Symanzik, J. (1997). Exploring Environmental Data in a Highly Immersive Virtual Reality Environment. *Inter. J. Environ. Monitoring and Assessment*, 51(1-2):441–450.
- Devroye, L. (1987). *A Course in Density Estimation*, Birkhäuser, Boston.
- Devroye, L. and Györfi, L. (1985), *Nonparametric Density Estimation: The L_1 View*, John Wiley, New York.
- Duda, R.O. and Hart, P.E. (1973). *Pattern Classification and Scene Analysis*, John Wiley, New York.
- Hall, P., Sheather, S.J., Jones, M.C. and Marron, J.S. (1991). On Optimal Data-Based Bandwidth Selection in Kernel Density Estimation. *Biometrika*, 78:263–270.
- Härdle, W. (1990). *Smoothing Techniques With Implementation in S*, Springer-Verlag, New York.

- Hazelton, M. (1996). Bandwidth Selection for Local Density Estimation. *Scandinavian Journal of Statistics*, 23:221–232.
- Inselberg, A. (1985). The Plane with Parallel Coordinates. *The Visual Computer*, 1:69–91.
- Jones, M.C., Marron, J.S., and Sheather, S.J. (1996). A Brief Survey of Bandwidth Selection for Density Estimation. *Journal of the American Statistical Association*, 91:401–407.
- Kronmal, R.A. and Tarter, M.E. (1968). The Estimation of Probability Densities and Cumulatives by Fourier Series Methods. *J. Amer. Statist. Assoc.*, 63:925–952.
- Loader, C. (1999). *Local Regression and Likelihood*, Springer, New York.
- Lorensen, W.E. and Cline, H.E. (1987). Marching Cubes: A High Resolution 3D Surface Construction Algorithm. *Computer Graphics*, 21:163–169.
- Minnotte, M.C. (1997). Nonparametric testing of the existence of modes. *The Annals of Statistics*, 25:1646–1660.
- Minnotte, M.C. and Scott, D.W. (1993). The mode tree: A tool for visualization of nonparametric density features. *Journal of Computational and Graphical Statistics*, 2:51–68.
- Nadaraya, E.A. (Kotz, S. Translator) (1989). *Nonparametric Estimation of Probability Densities and Regression Curves*, Kluwer Academic Publishers Group.
- Parzen, E. (1962). On Estimation of Probability Density Function and Mode. *Annals Math. Statist.*, 33:1065–1076.
- Prakasa Rao, B.L.S. (1983). *Nonparametric Functional Estimation*, Academic Press, Orlando, FL.
- Rice, J.A. (1984). Boundary Modification for Kernel Regression. *Commun. Statist.*, 13:893–900.
- Rosenblatt, M. (1956). Remarks on Some Nonparametric Estimates of a Density Function. *Ann. Math. Statist.*, 27:832–837.
- Rudemo, M. (1982). Empirical Choice of Histograms and Kernel Density Estimators. *Scandinavian Journal of Statistics*, 9:65–78.
- Sain, S.R., Baggerly, K.A., and Scott, D.W. (1994). Cross-Validation of Multivariate Densities. *Journal of the American Statistical Association*, 89:807–817.
- Sain, S.R. and Scott, D.W. (1996). On Locally Adaptive Density Estimation. *Journal of the American Statistical Association*, 91:1525–1534.
- Sain, S.R. and Scott, D.W. (2002). Zero-Bias Bandwidths for Locally Adaptive Kernel Density Estimation. *Scandinavian Journal of Statistics*, 29:441–460.
- Scott, D.W. (1979). On Optimal and Data-Based Histograms. *Biometrika*, 66:605–610.
- Scott, D.W. (1985a). On Optimal and Data-Based Frequency Polygons. *J. Amer. Statist. Assoc.*, 80:348–354.
- Scott, D.W. (1985b). Averaged Shifted Histograms: Effective Nonparametric Density Estimators in Several Dimensions. *Ann. Statist.*, 13:1024–1040.
- Scott, D.W. (1986). Data Analysis in 3 and 4 Dimensions with Nonparametric Density Estimation. In Wegman, E.J. and DePriest, D. (eds), *Statistical Image Processing and Graphics*, Marcel Dekker, New York:291–305.

- Scott, D.W. (1988). A Note on Choice of Bivariate Histogram Bin Shape. *J. of Official Statistics*, 4:47–51.
- Scott, D.W. (1992). *Multivariate Density Estimation: Theory, Practice, and Visualization*. John Wiley, New York.
- Scott, D.W. (2000). Multidimensional Smoothing and Visualization. In Schimek, M.G. (ed), *Smoothing and Regression. Approaches, Computation and Application*, John Wiley, New York:451–470.
- Scott, D.W. (2001). Parametric Statistical Modeling by Minimum Integrated Square Error. *Technometrics*, 43:274–285.
- Silverman, B.W. (1981). Using Kernel Density Estimates to Investigate Multimodality. *Journal of the Royal Statistical Society, Series B* 43:97–99.
- Silverman, B.W. (1982). Algorithm AS176. Kernel Density Estimation Using the Fast Fourier Transform, *Appl. Statist.* 31:93–99.
- Silverman, B.W. (1986). *Density Estimation for Statistics and Data Analysis*, Chapman and Hall, London.
- Silverman, B.W. and Jones, M.C. (1989). Fix, E. and Hodges, J. L. (1951): An important contribution to nonparametric discriminant analysis and density estimation: Commentary on Fix and Hodges (1951). *International Statistical Review*, 57:233–247.
- Simonoff, J.S. (1996). *Smoothing Methods in Statistics*, Springer-Verlag Inc.
- Sturges, H.A. (1926). The Choice of a Class Interval. *J. Amer. Statist. Assoc.*, 21:65–66.
- Swayne, D., Cook, D. and Buja, A. (1991). XGobi: Interactive Dynamic Graphics in the X Window System with a Link to S. *ASA Proceedings of the Section on Statistical Graphics*, pp. 1–8, ASA, Alexandria, VA.
- Tapia, R.A. and Thompson, J.R. (1978). *Nonparametric Probability Density Estimation* John Hopkins University Press, Baltimore.
- Tarter, M.E. (2000). *Statistical Curves and Parameters*, AK Peters, Natick, MA.
- Tarter, M.E. and Lock, M.D. (1993). *Model-Free Curve Estimation*, Chapman & Hall Ltd.
- Taylor, C.C. (1989). Bootstrap Choice of the Smoothing Parameter in Kernel Density Estimation. *Biometrika*, 76:705–712.
- Terrell, G.R. (1990). The Maximal Smoothing Principle in Density Estimation. *Journal of the American Statistical Association*, 85:470–477.
- Terrell, G.R. and Scott, D.W. (1980). On Improving Convergence Rates for Nonnegative Kernel Density Estimators. *Annals of Statistics*, 8:1160–1163.
- Terrell, G.R. and Scott, D.W. (1985). Oversmoothed Nonparametric Density Estimates. *Journal of the American Statistical Association*, 80:209–214.
- Terrell, G.R. and Scott, D.W. (1992). Variable Kernel Density Estimation. *The Annals of Statistics*, 20:1236–1265.
- Tufte, E.R. (1983). *The Visual Display of Quantitative Information*, Graphics Press, Cheshire, CT.
- Tukey, J.W. (1977). *Exploratory Data Analysis*, Addison-Wesley, Reading, MA.
- Wahba, G. (1981). Data-Based Optimal Smoothing of Orthogonal Series Density Estimates. *Ann. Statist.*, 9:146–156.

- Wainer, H. (1997). *Visual Revelations*, Springer-Verlag, New York.
- Walter, G. and Blum, J.R. (1979). Probability Density Estimation Using Delta Sequences. *Ann. Statist.*, 7:328–340.
- Wand, M.P. and Jones, M.C. (1995). *Kernel Smoothing*, Chapman & Hall Ltd.
- Wand, M.P., Marron, J.S. and Ruppert, D. (1991). Transformations in Density Estimation” *Journal of the American Statistical Association*, 86:343–353.
- Watson, G.S. (1969). Density Estimation by Orthogonal Series. *Ann. Math. Statist.*, 40:1496–1498.
- Wegman, E.J. (1990). Hyperdimensional Data Analysis Using Parallel Coordinates. *J. Amer. Statist. Assoc.*, 85:664–675.
- Wegman, E.J. and Depriest, D.J. (1986). *Statistical Image Processing and Graphics*, Marcel Dekker Inc, New York.
- Wertz, W. (1978). *Statistical Density Estimation: A Survey*, Vandenhoeck & Ruprecht, Göttingen.
- Wolff, R.S. and Yaeger, L. (1993). *Visualization of Natural Phenomena*, Springer-Verlag, New York.

Smoothing: Local Regression Techniques

III.5

Catherine Loader

5.1	<i>Smoothing</i>	540
5.2	<i>Linear Smoothing</i>	540
	Kernel Smoothers	541
	Local Regression.....	542
	Penalized Least Squares (Smoothing Splines).....	544
	Regression Splines.....	545
	Orthogonal Series	546
5.3	<i>Statistical Properties of Linear Smoothers</i>	547
	Bias	547
	Variance.....	549
	Degrees of Freedom	549
5.4	<i>Statistics for Linear Smoothers: Bandwidth Selection and Inference</i>	551
	Choosing Smoothing Parameters.....	551
	Normal-based Inference.....	554
	Bootstrapping	556
5.5	<i>Multivariate Smoothers</i>	557
	Two Predictor Variables.....	557
	Likelihood Smoothing	558
	Extensions of Local Likelihood.....	560

Smoothing methods attempt to find functional relationships between different measurements. As in the standard regression setting, the data is assumed to consist of measurements of a response variable, and one or more predictor variables. Standard regression techniques (Chap. III8.) specify a functional form (such as a straight line) to describe the relation between the predictor and response variables. Smoothing methods take a more flexible approach, allowing the data points themselves to determine the form of the fitted curve.

This article begins by describing several different approaches to smoothing, including kernel methods, local regression, spline methods and orthogonal series. A general theory of linear smoothing is presented, which allows us to develop methods for statistical inference, model diagnostics and choice of smoothing parameters.

The theory is then extended to more general settings, including multivariate smoothing and likelihood models.

5.1 Smoothing

Given a dataset consisting of several variables and multiple observations, the goal of smoothing is to construct a functional relationship among the variables.

The most common situation for smoothing is that of a classical regression setting, where one assumes that observations occur in (predictor, response) pairs. That is, the available data has the form

$$\{(x_i, Y_i) ; \quad i = 1, \dots, n\} ,$$

where x_i is a measurement of the predictor (or independent) variable, and Y_i is the corresponding response. A functional model relating the variables takes the form

$$Y_i = \mu(x_i) + \varepsilon_i , \quad (5.1)$$

where $\mu(x_i)$ is the mean function, and ε_i is a random error term. In classical regression analysis, one assumes a parametric form for the mean function; for example, $\mu(x) = a_0 + a_1x$. The problem of estimating the mean function then reduces to estimating the coefficients a_0 and a_1 .

The idea of smoothing methods is not to specify a parametric model for the mean function, but to allow the data to determine an appropriate functional form. Loosely stated, one assumes only that the mean function is smooth. Formal mathematical analysis may state the smoothness condition as a bound on derivatives of μ ; for example, $|\mu''(x)| \leq M$ for all x and a specified constant M .

Section 5.2 describes some of the most important smoothing methods. These all fall into a class of linear smoothers, and Sect. 5.3 develops important properties, including bias and variance. These results are applied to derive statistical procedures, including bandwidth selection, model diagnostics and goodness-of-fit testing in Sect. 5.4. Multivariate smoothing, when there are multiple predictor vari-

ables, is discussed in Sect. 5.5. Finally, Sect. 5.5.2 discusses extensions to likelihood smoothing.

Linear Smoothing

5.2

In this section, some of the most common smoothing methods are introduced and discussed.

Kernel Smoothers

5.2.1

The simplest of smoothing methods is a kernel smoother. A point x is fixed in the domain of the mean function $\mu(\cdot)$, and a smoothing window is defined around that point. Most often, the smoothing window is simply an interval $(x - h, x + h)$, where h is a fixed parameter known as the *bandwidth*.

The kernel estimate is a weighted average of the observations within the smoothing window:

$$\hat{\mu}(x) = \frac{\sum_{i=1}^n W\left(\frac{x_i - x}{h}\right) Y_i}{\sum_{j=1}^n W\left(\frac{x_j - x}{h}\right)}, \quad (5.2)$$

where $W(\cdot)$ is a weight function. The weight function is chosen so that most weight is given to those observations close to the fitting point x . One common choice is the bisquare function,

$$W(x) = \begin{cases} (1 - x^2)^2 & -1 \leq x \leq 1 \\ 0 & x > 1 \text{ or } x < -1 \end{cases}.$$

The kernel smoother can be represented as

$$\hat{\mu}(x) = \sum_{i=1}^n l_i(x) Y_i, \quad (5.3)$$

where the coefficients $l_i(x)$ are given by

$$l_i(x) = \frac{W\left(\frac{x_i - x}{h}\right)}{\sum_{j=1}^n W\left(\frac{x_j - x}{h}\right)}.$$

A *linear smoother* is a smoother that can be represented in the form (5.3) for appropriately defined weights $l_i(x)$. This linear representation leads to many nice statistical and computational properties, which will be discussed later.

The kernel estimate (5.2) is sometimes called the Nadaraya–Watson estimate (Nadaraya, 1964; Watson, 1964). Its simplicity makes it easy to understand and implement, and it is available in many statistical software packages. But its simplicity

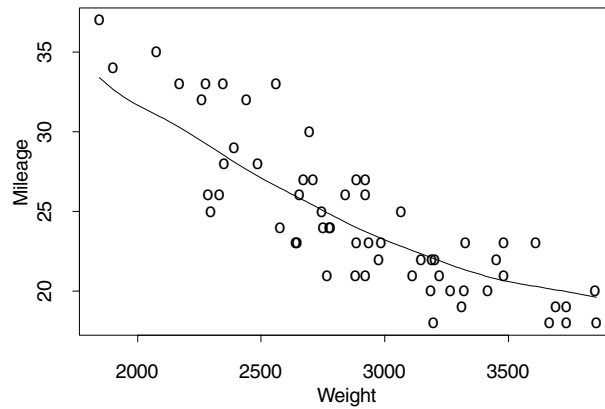


Figure 5.1. Kernel smooth of the fuel economy dataset. The bisquare kernel is used, with bandwidth $h = 600$ pounds

leads to a number of weaknesses, the most obvious of which is boundary bias. This can be illustrated through an example.

The fuel economy dataset consists of measurements of fuel usage (in miles per gallon) for sixty different vehicles. The predictor variable is the weight (in pounds) of the vehicle. Figure 5.1 shows a scatterplot of the sixty data points, together with a kernel smooth. The smooth is constructed using the bisquare kernel and bandwidth $h = 600$ pounds.

Over much of the domain of Fig. 5.1, the smooth fit captures the main trend of the data, as required. But consider the left boundary region; in particular, vehicles weighing less than 2200 pounds. All these data points lie *above* the fitted curve; the fitted curve will underestimate the economy of vehicles in this weight range. When the kernel estimate is applied at the left boundary (say, at Weight = 1800), all the data points used to form the average have Weight > 1800 , and correspondingly slope of the true relation induces boundary bias into the estimate.

More discussion of this and other weaknesses of the kernel smoother can be found in Hastie and Loader (1993). Many modified kernel estimates have been proposed, but one obtains more parsimonious solutions by considering alternative estimation procedures.

5.2.2 Local Regression

Local regression estimation was independently introduced in several different fields in the late nineteenth and early twentieth century (Henderson, 1916; Schiaparelli, 1866). In the statistical literature, the method was independently introduced from different viewpoints in the late 1970's (Cleveland, 1979; Katkovnik, 1979; Stone, 1977). Books on the topic include Fan and Gijbels (1996) and Loader (1999b).

The underlying principle is that a smooth function can be well approximated by a low degree polynomial in the neighborhood of any point x . For example, a local linear approximation is

$$\mu(x_i) \approx a_0 + a_1(x_i - x) \quad (5.4)$$

for $x - h \leq x_i \leq x + h$. A local quadratic approximation is

$$\mu(x_i) \approx a_0 + a_1(x_i - x) + \frac{a_2}{2}(x_i - x)^2 .$$

The local approximation can be fitted by locally weighted least squares. A weight function and bandwidth are defined as for kernel regression. In the case of local linear regression, coefficient estimates \hat{a}_0, \hat{a}_1 are chosen to minimize

$$\sum_{i=1}^n W \left(\frac{x_i - x}{h} \right) (Y_i - (a_0 + a_1(x_i - x)))^2 . \quad (5.5)$$

The local linear regression estimate is defined as

$$\hat{\mu}(x) = \hat{a}_0 . \quad (5.6)$$

Each local least squares problem defines $\hat{\mu}(x)$ at one point x ; if x is changed, the smoothing weights $W \left(\frac{x_i - x}{h} \right)$ change, and so the estimates \hat{a}_0 and \hat{a}_1 change.

Since (5.5) is a weighted least squares problem, one can obtain the coefficient estimates by solving the normal equations

$$\mathbf{X}^\top \mathbf{W} \left(\mathbf{Y} - \mathbf{X} \begin{pmatrix} \hat{a}_0 \\ \hat{a}_1 \end{pmatrix} \right) = 0 , \quad (5.7)$$

where \mathbf{X} is the *design matrix*:

$$\mathbf{X} = \begin{pmatrix} 1 & x_1 - x \\ \vdots & \vdots \\ 1 & x_n - x \end{pmatrix}$$

for local linear regression, \mathbf{W} is a diagonal matrix with entries $W \left(\frac{x_i - x}{h} \right)$ and $\mathbf{Y} = (Y_1 \ \dots \ Y_n)^\top$.

When $\mathbf{X}^\top \mathbf{W} \mathbf{X}$ is invertible, one has the explicit representation

$$\begin{pmatrix} \hat{a}_0 \\ \hat{a}_1 \end{pmatrix} = (\mathbf{X}^\top \mathbf{W} \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{W} \mathbf{Y} . \quad (5.8)$$

This shows that the local regression estimate is a linear estimate, as defined by (5.3). Explicitly, the coefficients $l_i(x)$ are given by

$$l(x)^\top = (l_1(x) \ \dots \ l_n(x)) = e_1^\top (\mathbf{X}^\top \mathbf{W} \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{W} , \quad (5.9)$$

where e_1^\top is the unit vector $(1 \ 0)$.

For local quadratic regression and higher order fits, one simply adds additional columns to the design matrix \mathbf{X} and vector e_1^\top .

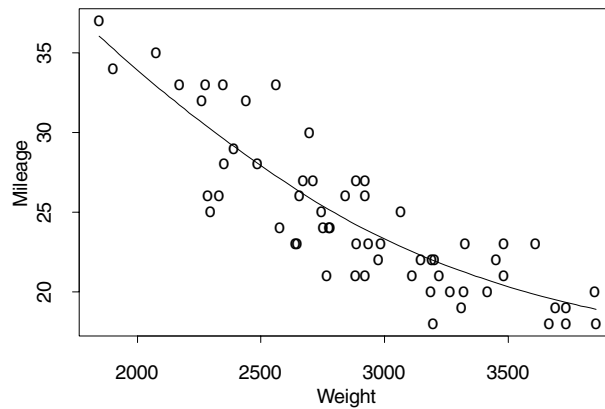


Figure 5.2. Local Linear Regression fitted to the fuel economy dataset. A bandwidth $h = 1000$ pounds is used

Figure 5.2 shows a local linear regression fit to the fuel economy dataset. This has clearly fixed the boundary bias problem observed in Fig. 5.1. With the reduction in boundary bias, it is also possible to substantially increase the bandwidth, from $h = 600$ pounds to $h = 1000$ pounds. As a result, the local linear fit is using much more data, meaning the estimate has less noise.

5.2.3 Penalized Least Squares (Smoothing Splines)

An entirely different approach to smoothing is through optimization of a penalized least squares criterion, such as

$$\sum_{i=1}^n (Y_i - \mu(x_i))^2 + \lambda \int \mu''(x)^2 dx, \quad (5.10)$$

where λ is specified constant. This criterion trades off fidelity to the data (measured by the residual sum-of-squares) versus roughness of the mean function (measured by the penalty term). The penalized least squares method chooses $\hat{\mu}$ from the class of twice differentiable functions to minimize the penalized least squares criterion.

The solution to this optimization problem is a piecewise polynomial, or spline function, and so penalized least squares methods are also known as smoothing splines. The idea was first considered in the early twentieth century (Whitaker, 1923). Modern statistical literature on smoothing splines began with work including Wahba and Wold (1975) and Silverman (1985). Books devoted to spline smoothing include Green and Silverman (1994) and Wahba (1990).

Suppose the data are ordered; $x_i \leq x_{i+1}$ for all i . Let $\hat{a}_i = \hat{\mu}(x_i)$, and $\hat{b}_i = \hat{\mu}'(x_i)$, for $i = 1, \dots, n$. Given these values, it is easy to show that between successive data points, $\hat{\mu}(x)$ must be the unique cubic polynomial interpolating these values:

$$\hat{\mu}(x) = a_i \phi_0(u) + b_i \Delta_i \psi_0(u) + a_{i+1} \phi_1(u) + b_{i+1} \Delta_i \psi_1(u),$$

where $\Delta_i = x_{i+1} - x_i$; $u = (x - x_i)/\Delta_i$ and

$$\phi_0(u) = 1 - u^2(3 - 2u)$$

$$\psi_0(u) = u(1 - u(2 - u))$$

$$\phi_1(u) = u^2(3 - 2u)$$

$$\psi_1(u) = u^2(u - 1).$$

Letting $\alpha^\top = (a_1 \ b_1 \ \dots \ a_n \ b_n)$, the penalty term $\int \mu''(x)^2 dx$ is a quadratic function of the parameters, and so (5.10) can be written as

$$\|Y - X\alpha\|^2 + \lambda\alpha^\top M\alpha,$$

for appropriate matrices M and X . The parameter estimates are given by

$$\hat{\alpha} = (X^\top X + \lambda M)^{-1} X^\top Y.$$

Figure 5.3 shows a smoothing spline fitted to the fuel economy dataset. Clearly, the fit is very similar to the local regression fit in Fig. 5.2. This situation is common for smoothing problems with a single predictor variable; with comparably chosen smoothing parameters, local regression and smoothing spline methods produce similar results. On the other hand, kernel methods can struggle to produce acceptable results, even on relatively simple datasets.

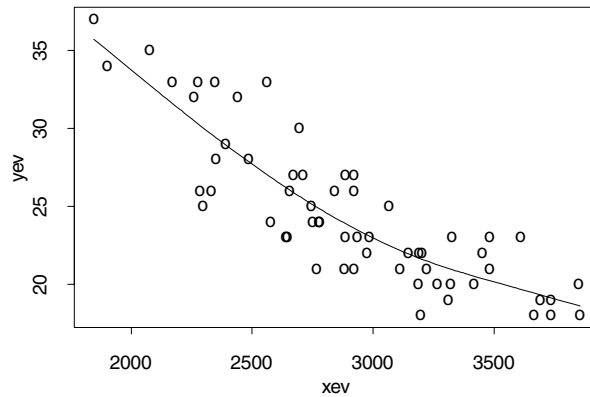


Figure 5.3. Smoothing Spline fitted to the fuel economy dataset. The penalty is $\lambda = 1.5 \times 10^8$ pounds³

Regression Splines

5.2.4

Regression splines begin by choosing a set of knots (typically, much smaller than the number of data points), and a set of basis functions spanning a set of piecewise polynomials satisfying continuity and smoothness constraints.

Let the knots be $v_1 < \dots < v_k$ with $v_1 = \min(x_i)$ and $v_k = \max(x_i)$. A linear spline basis is

$$f_j(x) = \begin{cases} \frac{x - v_{j-1}}{v_j - v_{j-1}} & v_{j-1} \leq x \leq v_j \\ \frac{v_{j+1} - x}{v_{j+1} - v_j} & v_j < x \leq v_{j+1} \\ 0 & \text{otherwise} \end{cases} ;$$

note that these functions span the space of piecewise linear functions with knots at v_1, \dots, v_k . The piecewise linear spline function is constructed by regressing the data onto these basis functions.

The linear spline basis functions have discontinuous derivatives, and so the resulting fit may have a jagged appearance. It is more common to use piecewise cubic splines, with the basis functions having two continuous derivatives. See Chap. 3 of Ruppert et al. (2003) for a more detailed discussion of regression splines and basis functions.

5.2.5 Orthogonal Series

Orthogonal series methods represent the data with respect to a series of orthogonal basis functions, such as sines and cosines. Only the low frequency terms are retained. The book Efromovich (1999) provides a detailed discussion of this approach to smoothing.

Suppose the x_i are equally spaced; $x_i = i/n$. Consider the basis functions

$$f_\omega(x) = a_\omega \cos(2\pi\omega x) ; \quad \omega = 0, 1, \dots, \lfloor n/2 \rfloor$$

$$g_\omega(x) = b_\omega \sin(2\pi\omega x) ; \quad \omega = 1, \dots, \lfloor (n-1)/2 \rfloor ,$$

where the constants a_ω, b_ω are chosen so that $\sum_{i=1}^n f_\omega(x_i)^2 = \sum_{i=1}^n g_\omega(x_i)^2 = 1$. Then the regression coefficients are

$$c_\omega = \sum_{i=1}^n f_\omega(x_i) Y_i$$

$$s_\omega = \sum_{i=1}^n g_\omega(x_i) Y_i$$

and the corresponding smooth estimate is

$$\hat{\mu}(x) = \sum_{\omega} h(\omega) (c_\omega f_\omega(x) + s_\omega g_\omega(x)) .$$

Here, $h(\omega)$ is chosen to ‘damp’ high frequencies in the observations; for example,

$$h(\omega) = \begin{cases} 1 & \omega \leq \omega_0 \\ 0 & \omega > \omega_0 \end{cases}$$

is a low-pass filter, passing all frequencies less than or equal to ω_0 .

Orthogonal series are widely used to model time series, where the coefficients c_ω and s_ω may have a physical interpretation: non-zero coefficients indicate the presence of cycles in the data. A limitation of orthogonal series approaches is that they are more difficult to apply when the x_i are not equally spaced.

Statistical Properties of Linear Smoothers

5.3

Each of the smoothing methods discussed in the previous section has one or more ‘smoothing parameters’ that control the amount of smoothing being performed. For example, the bandwidth h in the kernel smoother or local regression methods, and the parameter λ in the penalized likelihood criterion. In implementing the smoothers, the first question to be asked is how should the smoothing parameters be chosen? More generally, how can the performance of a smoother with given smoothing parameters be assessed? A deeper question is in comparing fits from different smoothers. For example, we have seen for the fuel economy dataset that a local linear fit with $h = 1000$ (Fig. 5.2) produces a fit similar to a smoothing spline with $\lambda = 1.5 \times 10^8$ (Fig. 5.3). Somehow, we want to be able to say these two smoothing parameters are equivalent.

As a prelude to studying methods for bandwidth selection and other statistical inference procedures, we must first study some of the properties of linear smoothers. We can consider measures of goodness-of-fit, such as the mean squared error,

$$\text{MSE}(x) = E((\hat{\mu}(x) - \mu(x))^2) = \text{var}(\hat{\mu}(x)) + \text{bias}(\hat{\mu}(x))^2,$$

where $\text{bias}(\hat{\mu}(x)) = E(\hat{\mu}(x)) - \mu(x)$.

Intuitively, as the bandwidth h increases, more data is used to construct the estimate $\hat{\mu}(x)$, and so the variance $\text{var}(\hat{\mu}(x))$ decreases. On the other hand, the local polynomial approximation is best over small intervals, so we expect the bias to increase as the bandwidth increases. Choosing h is a tradeoff between small bias and small variance, but we need more precise characterizations to derive and study selection procedures.

Bias

5.3.1

The bias of a linear smoother is given by

$$E(\hat{\mu}(x)) - \mu(x) = \sum_{i=1}^n l_i(x)E(Y_i) - \mu(x) = \sum_{i=1}^n l_i(x)\mu(x_i) - \mu(x). \quad (5.11)$$

As this depends on the unknown mean function $\mu(x)$, it is not very useful by itself, although it may be possible to estimate the bias by substituting an estimate for $\mu(x)$. To gain more insight, approximations to the bias are derived. The basic tools are

1. A low order Taylor series expansion of $\mu(\cdot)$ around the fitting point x .
2. Approximation of the sums by integrals.

For illustration, consider the bias of the local linear regression estimate defined by (5.6). A three-term Taylor series gives

$$\mu(x_i) = \mu(x) + (x_i - x)\mu'(x) + \frac{(x_i - x)^2}{2}\mu''(x) + o(h^2)$$

for $|x_i - x| \leq h$. Substituting this into (5.11) gives

$$\begin{aligned} E(\hat{\mu}(x)) - \mu(x) &= \mu(x) \sum_{i=1}^n l_i(x) + \mu'(x) \sum_{i=1}^n (x_i - x)l_i(x) \\ &\quad + \frac{\mu''(x)}{2} \sum_{i=1}^n (x_i - x)^2 l_i(x) - \mu(x) + o(h^2) . \end{aligned}$$

For local linear regression, it can be shown that

$$\begin{aligned} \sum_{i=1}^n l_i(x) &= 1 \\ \sum_{i=1}^n (x_i - x)l_i(x) &= 0 . \end{aligned}$$

This is a mathematical statement of the heuristically obvious property of the local linear regression: if data Y_i fall on a straight line, the local linear regression will reproduce that line. See Loader (1999b), p. 37, for a formal proof. With this simplification, the bias reduces to

$$E(\hat{\mu}(x)) - \mu(x) = \frac{\mu''(x)}{2} \sum_{i=1}^n (x_i - x)^2 l_i(x) + o(h^2) . \quad (5.12)$$

This expression characterizes the dependence of the bias on the mean function: the dominant term of the bias is proportional to the second derivative of the mean function.

The next step is to approximate summations by integrals, both in (5.12) and in the matrix equation (5.9) defining $l_i(x)$. This leads to

$$E(\hat{\mu}(x)) - \mu(x) \approx \mu''(x)h^2 \frac{\int v^2 W(v) dv}{2 \int W(v) dv} . \quad (5.13)$$

In addition to the dependence on $\mu''(x)$, we now see the dependence on h : as the bandwidth h increases, the bias increases quadratically with the bandwidth.

Bias expansions like (5.13) are derived much more generally by Ruppert and Wand (1994); their results cover arbitrary degree local polynomials and multi-dimensional fits also. Their results imply that when p , the degree of the local polynomial, is odd, the dominant term of the bias is proportional to $h^{p+1}\mu^{(p+1)}(x)$. When p is even, the first-order term can disappear, leading to bias of order h^{p+2} .

Variance

5.3.2

To derive the variance of a linear smoother, we need to make assumptions about the random errors ε_i in (5.1). The most common assumption is that the errors are independent and identically distributed, with variance $\text{var}(\varepsilon_i) = \sigma^2$. The variance of a linear smoother (5.3) is

$$\text{var}(\hat{\mu}(x)) = \sum_{i=1}^n l_i(x)^2 \text{var}(Y_i) = \sigma^2 \|l(x)\|^2. \quad (5.14)$$

As with bias, informative approximations to the variance can be derived by replacing sums by integrals. For local linear regression, this leads to

$$\text{var}(\hat{\mu}(x)) \approx \frac{\sigma^2}{nhf(x)} \frac{\int W(v)^2 dv}{\left(\int W(v) dv\right)^2}, \quad (5.15)$$

where $f(x)$ is the density of the design points x_i . The dependence on the sample size, bandwidth and design density through $1/(nhf(x))$ is universal, holding for any degree of local polynomial. The term depending on the weight function varies according to the degree of local polynomial, but generally increases as the degree of the polynomials increases. See Ruppert and Wand (1994) for details.

Degrees of Freedom

5.3.3

Under the model (5.1) the observation Y_i has variance σ^2 , while the estimate $\hat{\mu}(x_i)$ has variance $\sigma^2 \|l(x_i)\|^2$. The quantity $\|l(x_i)\|^2$ measures the variance reduction of the smoother at a data point x_i . At one extreme, if the 'smoother' interpolates the data, then $\hat{\mu}(x_i) = Y_i$ and $\|l(x_i)\|^2 = 1$. At the other extreme, if $\hat{\mu}(x_i) = \bar{Y}$, $\|l(x_i)\|^2 = 1/n$. Under mild conditions on the weight function, a local polynomial smoother satisfies

$$\frac{1}{n} \leq \|l(x_i)\|^2 \leq 1,$$

and $\|l(x_i)\|^2$ is usually a decreasing function of the bandwidth h .

A global measure of the amount of smoothing is provided by

$$v_2 = \sum_{i=1}^n \|l(x_i)\|^2.$$

This is one definition of the ‘degrees of freedom’ or ‘effective number of parameters’ of the smoother. It satisfies the inequalities

$$1 \leq \nu_2 \leq n .$$

An alternative representation of ν_2 is as follows. Let \mathbf{H} be the ‘hat matrix’, which maps the data to fitted values:

$$\begin{pmatrix} \hat{\mu}(x_1) \\ \vdots \\ \hat{\mu}(x_n) \end{pmatrix} = \mathbf{H}Y .$$

For a linear smoother, \mathbf{H} has rows $l(x_i)^\top$, and $\nu_2 = \text{trace}(\mathbf{H}^\top \mathbf{H})$.

The diagonal elements of \mathbf{H} , $l_i(x_i)$ provide another measure of the amount of smoothing at x_i . If the smooth interpolates the data, then $l(x_i)$ is the corresponding unit vector with $l_i(x_i) = 1$. If the smooth is simply the global average, $l_i(x_i) = 1/n$. The corresponding definition of degrees of freedom is

$$\nu_1 = \sum_{i=1}^n l_i(x_i) = \text{trace}(\mathbf{H}) .$$

For a least-squares fit, the hat matrix is a perpendicular projection operator, which is symmetric and idempotent. In this case, $\mathbf{H} = \mathbf{H}^\top \mathbf{H}$, and $\nu_1 = \nu_2$. For linear smoothers, the two definitions of degrees-of-freedom are usually not equal, but they are often of similar magnitude.

For the local linear regression in Fig. 5.2, the degrees of freedom are $\nu_1 = 3.54$ and $\nu_2 = 3.09$. For the smoothing spline smoother in Fig. 5.3, $\nu_1 = 3.66$ and $\nu_2 = 2.98$. By either measure the degrees of freedom are similar for the two fits. The degrees of freedom provides a mechanism by which different smoothers, with different smoothing parameters, can be compared: we simply choose smoothing parameters producing the same number of degrees of freedom. More extensive discussion of the degrees of freedom of a smoother can be found in Cleveland and Devlin (1988) and Hastie and Tibshirani (1990).

Variance Estimation. The final component needed for many statistical procedures is an estimate of the error variance σ^2 . One such estimate is

$$\hat{\sigma}^2 = \frac{1}{n - 2\nu_1 + \nu_2} \sum_{i=1}^n (Y_i - \hat{\mu}(x_i))^2 . \quad (5.16)$$

The normalizing constant is chosen so that if the bias of $\hat{\mu}(x_i)$ is neglected, $\hat{\sigma}^2$ is unbiased. See Cleveland and Devlin (1988).

Statistics for Linear Smoothers: Bandwidth Selection and Inference

5.4

We also want to perform statistical inference based on the smoothers. As for parametric regression, we want to construct confidence bands and prediction intervals based on the smooth curve. Given a new car that weighs 2800 pounds, what is its fuel economy? Tests of hypotheses can also be posed: for example, is the curvature observed in Fig. 5.2 significant, or would a linear regression be adequate? Given different classifications of car (compact, sporty, minivan etc.) is there differences among the categories that cannot be explained by weight alone?

Choosing Smoothing Parameters

5.4.1

All smoothing methods have one or more smoothing parameters: parameters that control the ‘amount’ of smoothing being performed. For example, the bandwidth h in the kernel and local regression estimates. Typically, bandwidth selection methods are based on an estimate of some goodness-of-fit criterion. Bandwidth selection is a special case of model selection, discussed more deeply in Chap. III.1.

How should smoothing parameters be used? At one extreme, there is full automation: optimization of the goodness-of-fit criterion produces a single ‘best’ bandwidth. At the other extreme is purely exploratory and graphical methods, using goodness-of-fit as a guide to help choose the best method.

Automation has the advantage that it requires much less work; a computer can be programmed to perform the optimization. But the price is a lack of reliability: fits with very different bandwidths can produce similar values of the goodness-of-fit criterion. The result is either high variability (producing fits that look under-smoothed) or high bias (producing fits that miss obvious features in the data).

Cross Validation. Cross validation (CV) focuses on the prediction problem: if the fitted regression curve is used to predict new observations, how good will the prediction be? If a new observation is made at $x = x_0$, and the response Y_0 is predicted by $\hat{Y}_0 = \hat{\mu}(x_0)$, what is the prediction error? One measure is

$$E((Y_0 - \hat{Y}_0)^2) .$$

The method of CV can be used to estimate this quantity. In turn, each observation (x_i, Y_i) is omitted from the dataset, and is ‘predicted’ by smoothing the remaining $n - 1$ observations. This leads to the CV score

$$CV(\hat{\mu}) = \frac{1}{n} \sum_{i=1}^n (Y_i - \hat{\mu}_{-i}(x_i))^2 , \quad (5.17)$$

where $\hat{\mu}_{-i}(\cdot)$ denotes the smoothed estimate when the single data point (x_i, Y_i) are omitted from the dataset; only the remaining $n - 1$ data points are used to compute the estimate.

Formally computing each of the leave-one-out regression estimates $\hat{\mu}_{-i}(\cdot)$ would be highly computational, and so at a first glance computation of the CV score (5.17) looks prohibitively expensive. But there is a remarkable simplification, valid for nearly all common linear smoothers (and all those discussed in Sect. 5.2):

$$\hat{\mu}_{-i}(x_i) = \frac{\hat{\mu}(x_i) - l_i(x_i)Y_i}{1 - l_i(x_i)}.$$

With this simplification, the CV criterion becomes

$$\text{CV}(\hat{\mu}) = \frac{1}{n} \sum_{i=1}^n \frac{(Y_i - \hat{\mu}(x_i))^2}{(1 - l_i(x_i))^2}.$$

Generalized cross validation (GCV) replaces each of the influence values $l_i(x_i)$ by the average, v_1/n . This leads to

$$\text{GCV}(\hat{\mu}) = n \frac{\sum_{i=1}^n (Y_i - \hat{\mu}(x_i))^2}{(n - v_1)^2}.$$

Figure 5.4 shows the GCV scores for the fuel economy dataset, and using kernel and local linear smoothers with a range of bandwidths. Note the construction of the plot: the fitted degrees of freedom v_1 are used as the x axis. This allows us to meaningfully superimpose and compare the GCV curves arising from different smoothing methods. From right to left, the points marked '0' represent a kernel smoother with $h = 300, 400, 500, 600, 800$ and 1000 , and points marked '1' represent a local linear smoother with $h = 400, 500, 700, 1000, 1500, 2000$ and ∞ .

The interpretation of Fig. 5.4 is that for any fixed degrees of freedom, the local linear fit outperforms the kernel fit. The best fits obtained are the local linear, with 3 to 3.5 degrees of freedom, or h between 1000 and 1500.

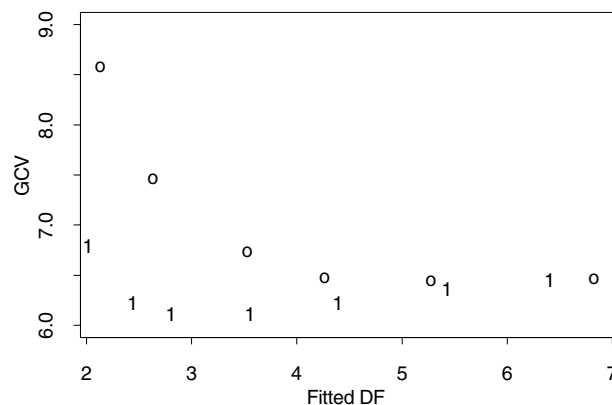


Figure 5.4. GCV scores for the fuel economy dataset. Points marked 0 are for kernel smoothers with a range of bandwidths h , and points marked 1 are for a local linear smoother

Unbiased Risk Estimation. A risk function measures the distance between the true regression function and the estimate; for example,

$$R(\mu, \hat{\mu}) = \frac{1}{\sigma^2} \sum_{i=1}^n E((\hat{\mu}(x_i) - \mu(x_i))^2) . \quad (5.18)$$

Ideally, a good estimate would be one with low risk. But since μ is unknown, $R(\mu, \hat{\mu})$ cannot be evaluated directly.

Instead, the risk must be estimated. An unbiased estimate is

$$\hat{R}(\mu, \hat{\mu}) = \frac{1}{\sigma^2} \sum_{i=1}^n (Y_i - \hat{\mu}(x_i))^2 - n + 2\nu_1$$

(Mallows, 1973; Cleveland and Devlin, 1988). The unbiased risk estimate is equivalent to Akaike's Information Criterion (Akaike, 1972, 1974). To implement the unbiased risk estimate one needs to substitute an estimate for σ^2 ; Cleveland and Devlin recommend using (5.16) with a small bandwidth.

The unbiased risk estimate can be used similarly to GDV. One computes $\hat{R}(\mu, \hat{\mu})$ for a range of different fits $\hat{\mu}$, and plots the resulting risk estimates versus the degrees of freedom. Fits producing a small risk estimate are considered best.

Bias Estimation and Plug-in Methods. An entirely different class of bandwidth selection methods, often termed plug-in methods, attempt to directly estimate a risk measure by estimating the bias and variance. The method has been developed mostly in the context of kernel density estimation, but adaptations to kernel regression and local polynomial regression can be found in Fan and Gijbels (1995) and Ruppert et al. (1995).

Again focusing on the squared-error risk, we have the bias-variance decomposition

$$\begin{aligned} \sigma^2 R(\mu, \hat{\mu}) &= \sum_{i=1}^n \text{bias}(\hat{\mu}(x_i))^2 + \sum_{i=1}^n \text{var}(\hat{\mu}(x_i)) \\ &= \sum_{i=1}^n \left(\sum_{j=1}^n l_j(x_i) \mu(x_j) - \mu(x_i) \right)^2 + \sigma^2 \sum_{i=1}^n \|l(x_i)\|^2 . \end{aligned} \quad (5.19)$$

A plug-in estimate begins by constructing a preliminary *pilot estimate* of the mean function $\mu(\cdot)$. This is then substituted into the risk estimate (5.19), which can then be minimized over the bandwidth h .

There are many variants of the plug-in idea in the statistics literature. Most simplify the risk function using asymptotic approximations such as (5.13) and (5.15) for the bias and variance; making these substitutions in (5.19) gives

$$\sigma^2 R(\mu, \hat{\mu}) \approx h^4 \left(\frac{\int v^2 W(v) dv}{2 \int W(v) dv} \right)^2 \sum_{i=1}^n \mu''(x_i)^2 + \frac{\sigma^2}{nh} \frac{\int W(v)^2 dv}{(\int W(v) dv)^2} \sum_{i=1}^n \frac{1}{f(x_i)} .$$

If the design points are uniformly distributed on an interval $[a, b]$ say, then approximating the sums by integrals gives

$$\sigma^2 R(\mu, \hat{\mu}) \approx nh^4 \left(\frac{\int v^2 W(v) dv}{2 \int W(v) dv} \right)^2 \frac{1}{b-a} \int_a^b \mu''(x)^2 dx + \frac{(b-a)\sigma^2}{h} \frac{\int W(v)^2 dv}{(\int W(v) dv)^2}.$$

Minimizing this expression over h yields an asymptotically optimal bandwidth:

$$h_{\text{opt}}^5 = \frac{\sigma^2 (b-a)^2 \int W(v)^2 dv}{n \left(\int v^2 W(v) dv \right)^2 \int_a^b \mu''(x)^2 dx}.$$

Evaluation of h_{opt} requires substitution of estimates for $\int_a^b \mu''(x)^2 dx$ and of σ^2 . The estimate (5.16) can be used to estimate σ^2 , but estimating $\int_a^b \mu''(x)^2 dx$ is more problematic. One technique is to estimate the second derivative using a 'pilot' estimate of the smooth, and then use the estimate

$$\int_a^b \hat{\mu}''(x)^2 dx.$$

If a local quadratic estimate is used at the pilot stage, the curvature coefficient \hat{a}_2 can be used as an estimate of $\mu''(x)$.

But the use of a pilot estimate to estimate the second derivative is problematic. The pilot estimate itself has a bandwidth that has to be selected, and the estimated optimal bandwidth \hat{h}_{opt} is highly sensitive to the choice of pilot bandwidth. Roughly, if the pilot estimate smooths out important features of μ , so will the estimate $\hat{\mu}$ with bandwidth \hat{h}_{opt} . More discussion of this point may be found in Loader (1999a).

5.4.2 Normal-based Inference

Inferential procedures for smoothers include the construction of confidence bands for the true mean function, and procedures to test the adequacy of simpler models. In this section, some of the main ideas are briefly introduced; more extensive discussion can be found in the books Azzalini and Bowman (1997), Härdle (1990), Hart (1997) and Loader (1999b).

Confidence Intervals. If the errors ε_i are normally distributed, then confidence intervals for the true mean can be constructed as

$$\hat{\mu}(x) \pm c\hat{\sigma}\|l(x)\|.$$

The constant c can be chosen from the Student's t distribution with degrees of freedom equal to $n - 2\nu_1 + \nu_2$ (alternative choices are discussed below in the context of testing). These confidence intervals are pointwise intervals for $E(\hat{\mu}(x))$:

$$P(|\hat{\mu}(x) - E(\hat{\mu}(x))| < c\hat{\sigma}\|l(x)\|) = 1 - \alpha.$$

To construct confidence intervals for $\mu(x)$, one must either choose the bandwidth sufficiently small so that the bias can be ignored, or explicitly estimate the bias. The latter approach suffers from the same weaknesses observed in plug-in bandwidth selection.

Tests of Hypothesis. Consider the problem of testing for the adequacy of a linear model. For example, in the fuel economy dataset of Figs. 5.1 and 5.2, one may be interested in knowing whether a linear regression, $\mu(x) = a + bx$ is adequate, or alternatively whether the departure from linearity indicated by the smooth is significant. This hypothesis testing problem can be stated as

$$H_0 : \mu(x) = a + bx \quad \text{for some } a, b$$

$$H_1 : \text{otherwise .}$$

In analogy with the theory of linear models, an F ratio can be formed by fitting both the null and alternative models, and considering the difference between the fits. Under the null model, parametric least squares is used; the corresponding fitted values are MY where M is the hat matrix for the least squares fit. Under the alternative model, the fitted values are HY , where H is the hat matrix for a local linear regression. An F ratio can then be formed as

$$F = \frac{\|HY - MY\|^2/v}{\hat{\sigma}^2},$$

where $v = \text{trace}((H - M)^\top(H - M))$.

What is the distribution of F when H_0 is true? Since H is not a perpendicular projection operator, the numerator does not have a χ^2 distribution, and F does not have an exact F distribution. None-the-less, we can use an approximating F distribution. Based on a one-moment approximation, the degrees of freedom are v and $n - 2v_1 + v_2$.

Better approximations are obtained using the two-moment Satterwaite approximation, as described in Cleveland and Devlin (1988). This method matches both the mean and variance of chi-square approximations to the numerator and denominator. Letting $\Lambda = (H - M)^\top(H - M)$, the numerator degrees of freedom for the F distribution are given by $\text{trace}(\Lambda)^2/\text{trace}(\Lambda^2)$. A similar adjustment is made to the denominator degrees of freedom. Simulations reported in Cleveland and Devlin (1988) suggest the two-moment approximation is adequate for setting critical values.

For the fuel economy dataset, we obtain $F = 7.247$, $v = 1.0866$ and $n - 2v_1 + v_2 = 55.997$. Using the one-moment approximation, the p -value is 0.0079. The two-moment approximation gives a p -value of 0.0019. Both methods indicate that the nonlinearity is significant, although there is some discrepancy between the P -values.

5.4.3 Bootstrapping

The F -tests in the previous section are approximate, even when the errors ε_i are normally distributed. Additionally, the degrees-of-freedom computations (particularly for the two-moment approximation) require $O(n^3)$ computations, which is prohibitively expensive for n more than a few hundred.

An alternative to the F approximations is to simulate the null distribution of the F ratio. A bootstrap method (Chap. III.2) performs the simulations using the empirical residuals to approximate the true error distribution:

- Let $r_i = Y_i - \hat{\mu}(x_i)$.
- Resample: $Y_i^* = \hat{\mu}(x_i) + \varepsilon_i^*$, $i = 1, \dots, n$, where ε_i^* is drawn from r_1, \dots, r_n .
- Compute the F statistic based on the resampled data:

$$F^* = \frac{\|HY^* - MY^*\|^2/\nu}{(\hat{\sigma}^*)^2}.$$

This procedure is repeated a large number of times (say $B = 1000$) and tabulation of the resulting F^* values provides an estimate of the true distribution of the F ratio.

Remark. Since the degrees of freedom do not change with the replication, there is no need to actually compute the normalizing constant. Instead, one can simply work with the modified F ratio,

$$F_B = \frac{\|HY^* - MY^*\|^2}{\|(I - H)Y^*\|^2}.$$

Figure 5.5 compares the bootstrap distribution of the F ratio and the 1 and 2 moment F approximations for the fuel economy dataset. The bootstrap method uses 10,000 bootstrap replications, and the density is estimated using the Local Likelihood method (Sect. 5.5.2 below). Except at the left end-point, there is generally good agreement between the bootstrap density and the two-moment density. The

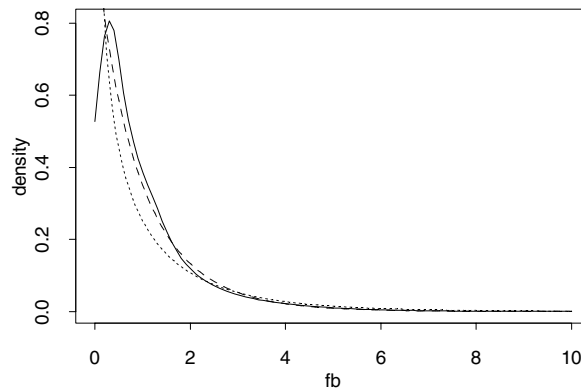


Figure 5.5. Estimated density of the F ratio, based on the bootstrap method (solid line); 1-moment F approximation (short dashed line) and 2-moment F approximation (long dashed line)

upper 5% quantiles are 3.21 based on the two-moment approximation, and 3.30 based on the bootstrap sample. The one-moment approximation has a critical value of 3.90. Based on the observed $F = 7.248$, the bootstrap p -value is 0.0023, again in close agreement with the two-moment method.

Multivariate Smoothers

5.5

When there are multiple predictor variables, the smoothing problem becomes multivariate: $\mu(x)$ is now a surface. The definition of kernel and local regression smoothers can be extended to estimate a regression surface with any number of predictor variables, although the methods become less useful for more than 2 or 3 variables. There are several reasons for this:

- Data sparsity – the curse of dimensionality.
- Visualization issues – how does one view and interpret a high dimensional smooth regression surface?
- Computation is often much more expensive in high dimensions.

For these reasons, use of local polynomials and other smoothers to model high dimensional surfaces is rarely recommended, and the presentation here is restricted to the two-dimensional case. In higher dimensions, smoothers can be used in conjunction with dimension reduction procedures (Chap. III.6), which attempt to model the high-dimensional surface through low-dimensional components. Examples of this type of procedure include Projection Pursuit (Friedman and Stuetzle, 1981), Additive Models (Hastie and Tibshirani, 1990), Semiparametric Models (Ruppert et al. (2003) and Chap. III.10) and recursive partitioning (Chap. III.14).

Two Predictor Variables

5.5.1

Suppose the dataset consists of n vectors (u_i, v_i, Y_i) , where u_i and v_i are considered predictor variables, and Y_i is the response. For simplicity, we'll use $x_i = \begin{pmatrix} u_i & v_i \end{pmatrix}^\top$ to denote a vector of the predictor variables. The data are modeled as

$$Y_i = \mu(u_i, v_i) + \varepsilon_i = \mu(x_i) + \varepsilon_i .$$

Bivariate smoothers attempt to estimate the surface $\mu(u_i, v_i)$. Kernel and local regression methods can be extended to the bivariate case, simply by defining smoothing weights on a plane rather than on a line. Formally, a bivariate local regression estimate at a point $x = (u, v)^\top$ can be constructed as follows:

1. Define a distance measure $\varphi(x, x_i)$ between the data points and fitting point. A common choice is Euclidean distance,

$$\varphi(x, x_i) = \sqrt{(u_i - u)^2 + (v_i - v)^2} .$$

2. Define the smoothing weights using a kernel function and bandwidth:

$$w_i(x) = W\left(\frac{\varrho(x, x_i)}{h}\right).$$

3. Define a local polynomial approximation, such as a local linear approximation

$$\mu(u_i, v_i) \approx a_0 + a_1(u_i - u) + a_2(v_i - v)$$

when (u_i, v_i) is close to (u, v) . More generally, a local polynomial approximation can be written

$$\mu(x_i) \approx \langle a, A(x_i - x) \rangle,$$

where a is a vector of coefficients, and $A(\cdot)$ is a vector of basis polynomials.

4. Estimate the coefficient vector by local least squares. That is, choose \hat{a} to minimize

$$\sum_{i=1}^n w_i(x) (Y_i - \langle a, A(x_i - x) \rangle)^2.$$

5. The local polynomial estimate is then

$$\hat{\mu}(x) = \hat{a}_0.$$

5.5.2 Likelihood Smoothing

A likelihood smoother replaces the model (5.1) with a distributional assumption

$$Y_i \sim f(y, \mu_i),$$

where $f(y, \mu)$ is a specified family of densities, parameterized so that $E(Y_i) = \mu_i$. The family may be chosen depending on the response variable. If Y_i is a count, then the Poisson family is a natural choice:

$$f(y, \mu) = \frac{\mu^y e^{-\mu}}{y!}; \quad y = 0, 1, 2, \dots$$

If Y_i is a 0/1 (or no/yes) response, then the Bernoulli family is appropriate:

$$f(y, \mu) = \mu^y (1 - \mu)^{1-y}; \quad y = 0, 1.$$

Given the data, the log-likelihood is

$$\mathcal{L}(\mu_1, \dots, \mu_n) = \sum_{i=1}^n \log f(Y_i, \mu_i).$$

The goal is to estimate the mean function, $\mu_i = \mu(x_i)$ for an observed set of covariates x_i . A generalized linear model (Chap. III.7) uses a parametric model for the mean function. Likelihood smoothers assume only that the mean is a smooth function of the covariates.

The earliest work on likelihood smoothing is Henderson (1924), who used a penalized binomial likelihood to estimate mortality rates. The local likelihood method described below can be viewed as an extension of local polynomial regression, and was introduced by Tibshirani and Hastie (1987).

Local Likelihood Estimation. Local likelihood estimation is based on a locally weighted version of the log-likelihood:

$$\mathcal{L}_x(\mu_1, \dots, \mu_n) = \sum_{i=1}^n w_i(x) \log f(Y_i, \mu_i) .$$

A local polynomial approximation is then used for a transformation of the mean function. For example, a local quadratic approximation is

$$\begin{aligned} \theta(x_i) &= g(\mu(x_i)) \\ &\approx a_0 + a_1(x_i - x) + \frac{a_2}{2}(x_i - x)^2 . \end{aligned}$$

The function $g(\mu)$ is the link function. Its primary goal is to remove constraints on the mean by mapping the parameter space to $(-\infty, \infty)$. For example, in the Poisson case, the parameter space is $0 < \mu < \infty$. If the log transformation $\theta = \log(\mu)$ is used, then the parameter space becomes $-\infty < \theta < \infty$.

Let $l(y, \theta) = \log f(y, \mu)$ where $\theta = g(\mu)$, so that the locally weighted log-likelihood becomes

$$\mathcal{L}_x = \sum_{i=1}^n w_i(x) l(Y_i, \theta(x_i)) .$$

The maximizer satisfies the likelihood equations,

$$\sum_{i=1}^n w_i(x) \begin{pmatrix} 1 \\ x_i - x \\ \frac{1}{2}(x_i - x)^2 \end{pmatrix} \dot{l}(Y_i, \theta(x_i)) = 0 , \quad (5.20)$$

where

$$\dot{l} = \frac{\partial}{\partial \theta} l(y, \theta) .$$

In matrix notation, this system of equations can be written in a form similar to (5.7):

$$\mathbf{X}^\top \mathbf{W} \dot{l}(Y, \mathbf{X}a) = 0 . \quad (5.21)$$

This system of equations is solved to find parameter estimates \hat{a}_0, \hat{a}_1 and \hat{a}_2 . The local likelihood estimate is defined as

$$\hat{\mu}(x) = g^{-1}(\hat{a}_0) .$$

Solving the Local Likelihood Equations. The local likelihood equations (5.20) are usually non-linear, and so the solution must be obtained through iterative methods. The Newton–Raphson updating formula is

$$\hat{a}^{(j+1)} = \hat{a}^{(j)} + (\mathbf{X}^\top \mathbf{W} \mathbf{V} \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{W} l(Y, \mathbf{X} \hat{a}^{(j)}) , \quad (5.22)$$

where \mathbf{V} is a diagonal matrix with entries

$$-\frac{\partial^2}{\partial \theta^2} l(y, \theta) .$$

For many common likelihoods $l(Y, \theta)$ is concave. Under mild conditions on the design points, this implies that the local likelihood is also concave, and has a unique global maximizer. If the Newton–Raphson algorithm converges, it must converge to this global maximizer.

The Newton–Raphson algorithm (5.22) cannot be guaranteed to converge from arbitrary starting values. But for concave likelihoods, $\hat{a}^{(j+1)} - \hat{a}^{(j)}$ is guaranteed to be an ascent direction, and convergence can be ensured by controlling the step size.

Statistics for the Local Likelihood Estimate. Since the local likelihood estimate does not have an explicit representation, statistical properties cannot be derived as easily as in the local regression case. But a Taylor series expansion of the local likelihood gives an approximate linearization of the estimate, leading to theory parallel to that developed in Sects. 5.3 and 5.4 for local regression. See Chap. 4 of Loader (1999b).

5.5.3 Extensions of Local Likelihood

The local likelihood method has been formulated for regression models. But variants of the method have been derived for numerous other settings, including robust regression, survival models, censored data, proportional hazards models, and density estimation. References include Tibshirani and Hastie (1987), Hjort and Jones (1996), Loader (1996, 1999b).

Robust Smoothing. Robust smoothing combines the ideas of robust estimation (Chap. III.9) with smoothing. One method is local M-estimation: choose \hat{a} to minimize

$$\sum_{i=1}^n w_i(x) \rho(Y_i - \langle a, A(x_i - x) \rangle) ,$$

and estimate $\hat{\mu}(x) = \hat{a}_0$. If $\varrho(u) = u^2$, this corresponds to local least squares estimation. If $\varrho(u)$ is a symmetric function that increases more slowly than u^2 , then the resulting estimate is more robust to outliers in the data. One popular choice of $\varrho(u)$ is the Huber function:

$$\varrho(u) = \begin{cases} u^2 & |u| \leq c \\ c(2|u| - c) & |u| > c \end{cases}.$$

References include Härdle (1990) and Loader (1999b). Another variant of M-estimation for local regression is the iterative procedure of Cleveland (1979).

Density Estimation. Suppose X_1, \dots, X_n are an independent sample from a density $f(x)$. The goal is to estimate $f(x)$. The local likelihood for this problem is

$$\mathcal{L}_x(a) = \sum_{i=1}^n w_i(x) \langle a, A(x_i - x) \rangle - n \int_{\mathcal{X}} W\left(\frac{u-x}{h}\right) e^{(a, A(u-x))} du.$$

Letting \hat{a} be the maximizer of the local log-likelihood, the local likelihood estimate is $\hat{f}(x) = \exp(\hat{a}_0)$. See Hjort and Jones (1996) and Loader (1996).

The density estimation problem is discussed in detail, together with graphical techniques for visualizing densities, in Chap. III.4.

Acknowledgements. This work was supported by National Science Foundation Grant DMS 0306202.

References

- Akaike, H. (1972). Information theory and an extension of the maximum likelihood principle. In Petrov, B.N. and Csàki, F., editors, *Second International Symposium on Information Theory*, pages 267–281, Budapest. Akademia Kiadó.
- Akaike, H. (1974). A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, 19:716–723.
- Azzalini, A. and Bowman, A.W. (1997). *Applied Smoothing Techniques for Data Analysis*. Oxford University Press, Oxford.
- Cleveland, W.S. (1979). Robust locally weighted regression and smoothing scatterplots. *Journal of the American Statistical Association*, 74:829–836.
- Cleveland, W.S. and Devlin, S.J. (1988). Locally weighted regression: An approach to regression analysis by local fitting. *Journal of the American Statistical Association*, 83:596–610.
- Efromovich, S. (1999). *Nonparametric Curve Estimation*. Springer, New York.
- Fan, J. and Gijbels, I. (1995). Data-driven bandwidth selection in local polynomial fitting: variable bandwidth and spatial adaptation. *Journal of the Royal Statistical Society, Series B*, 57:371–394.

- Fan, J. and Gijbels, I. (1996). *Local Polynomial Modelling and its Applications*. Chapman and Hall, London.
- Friedman, J. and Stuetzle, W. (1981). Projection pursuit regression. *Journal of the American Statistical Association*, 76:817–823.
- Green, P.J. and Silverman, B. (1994). *Nonparametric Regression and Generalized Linear Models: A roughness penalty approach*. Chapman and Hall, London.
- Härdle, W. (1990). *Applied Nonparametric Regression*. Cambridge University Press, Cambridge.
- Hart, J.D. (1997). *Nonparametric Smoothing and Lack-of-Fit Tests*. Springer, New York.
- Hastie, T.J. and Loader, C.R. (1993). Local regression: Automatic kernel carpentry (with discussion). *Statistical Science*, 8:120–143.
- Hastie, T.J. and Tibshirani, R.J. (1990). *Generalized Additive Models*. Chapman and Hall, London.
- Henderson, R. (1916). Note on graduation by adjusted average. *Transactions of the Actuarial Society of America*, 17:43–48.
- Henderson, R. (1924). A new method of graduation. *Transactions of the Actuarial Society of America*, 25:29–40.
- Hjort, N.L. and Jones, M.C. (1996). Locally parametric nonparametric density estimation. *The Annals of Statistics*, 24:1619–1647.
- Katkovnik, V.Y. (1979). Linear and nonlinear methods of nonparametric regression analysis. *Soviet Automatic Control*, 5:35–46 (25–34).
- Loader, C. (1996). Local likelihood density estimation. *The Annals of Statistics*, 24:1602–1618.
- Loader, C. (1999a). Bandwidth selection: Classical or plug-in? *The Annals of Statistics*, 27:415–438.
- Loader, C. (1999b). *Local Regression and Likelihood*. Springer, New York.
- Mallows, C.L. (1973). Some comments on c_p . *Technometrics*, 15:661–675.
- Nadaraya, E.A. (1964). On estimating regression. *Theory of Probability and its Applications*, 9:157–159 (141–142).
- Ruppert, D., Sheather, S.J., and Wand, M.P. (1995). An effective bandwidth selector for local least squares regression. *Journal of the American Statistical Association*, 90:1257–1270.
- Ruppert, D. and Wand, M.P. (1994). Multivariate locally weighted least squares regression. *The Annals of Statistics*, 22:1346–1370.
- Ruppert, D., Wand, M.P., and Carroll, R.J. (2003). *Semiparametric Regression*. Cambridge University Press, Cambridge.
- Schiaparelli, G.V. (1866). Sul modo di ricavare la vera espressione delle leggi delta natura dalle curve empiricae. *Effemeridi Astronomiche di Milano per l'Arno*, 857:3–56.
- Silverman, B.W. (1985). Some aspects of the spline smoothing approach to nonparametric regression curve fitting (with discussion). *Journal of the Royal Statistical Society, Series B*, 47:1–52.
- Stone, C.J. (1977). Consistent nonparametric regression (with discussion). *The Annals of Statistics*, 5:595–645.

- Tibshirani, R.J. and Hastie, T.J. (1987). Local likelihood estimation. *Journal of the American Statistical Association*, 82:559–567.
- Wahba, G. (1990). *Spline Models for Observational Data*. SIAM, Philadelphia.
- Wahba, G. and Wold, S. (1975). A completely automatic French curve: Fitting spline functions by cross-validation. *Communications in Statistics*, 4:1–17.
- Watson, G.S. (1964). Smooth regression analysis. *Sankhya Series A*, 26:359–372.
- Whitaker, E.T. (1923). On a new method of graduation. *Proceedings of the Edinburgh Mathematical Society*, 41:62–75.

Dimension Reduction Methods III.6

Masahiro Mizuta

6.1	<i>Introduction</i>	566
6.2	<i>Linear Reduction of High-dimensional Data</i>	566
	Principal Component Analysis	566
	Projection Pursuit	568
6.3	<i>Nonlinear Reduction of High-dimensional Data</i>	571
	Generalized Principal Component Analysis	571
	Algebraic Curve and Surface Fitting	575
	Principal Curves.....	581
6.4	<i>Linear Reduction of Explanatory Variables</i>	584
6.5	<i>Concluding Remarks</i>	588

6.1 Introduction

One characteristic of computational statistics is the processing of enormous amounts of data. It is now possible to analyze large amounts of high-dimensional data through the use of high-performance contemporary computers. In general, however, several problems occur when the number of dimensions becomes high. The first problem is an explosion in execution time. For example, the number of combinations of subsets taken from p variables is 2^p ; when p exceeds 20, calculation becomes difficult pointing terms of computation time. When p exceeds 25, calculation becomes an impossible no matter what type of computer is used. This is a fundamental situation that arises in the selection of explanatory variables during regression analysis. The second problem is the sheer cost of surveys or experiments. When questionnaire surveys are conducted, burden is placed on the respondent because there are many questions. And since there are few inspection items to a patient, there are few the burdens on the body or on cost. The third problem is the essential restriction of methods. When the number of explanatory variables is greater than the data size, most methods are incapable of directly dealing with the data; microarray data are typical examples of this type of data.

For these reasons, methods for dimension reduction without loss of statistical information are important techniques for data analysis. In this chapter, we will explain linear and nonlinear methods for dimension reduction; linear methods reduce dimension through the use of linear combinations of variables, and nonlinear methods do so with nonlinear functions of variables. We will also discuss the reduction of explanatory variables in regression analysis. Explanatory variables can be reduced with several linear combinations of explanatory variables.

Linear Reduction of High-dimensional Data

6.2

The p -dimensional data can be reduced into q -dimensional data using q linear combinations of p variables. The linear combinations can be considered as linear projection. Most methods for reduction involve the discovery of linear combinations of variables under set criterion. Principal component analysis (PCA) and projection pursuit are typical methods of this type. These methods will be described in the following subsections.

6.2.1 Principal Component Analysis

Suppose that we have observations of p variables size n ; $\{x_i; i = 1, 2, \dots, n\}$ (referred to as X hereafter). PCA is conducted for the purpose of constructing linear combinations of variables so that their variances are large under certain conditions.

A linear combination of variables is denoted by $\{\mathbf{a}^\top \mathbf{x}_i; i = 1, 2, \dots, n\}$ (simply, $\mathbf{a}^\top \mathbf{X}$), where $\mathbf{a} = (a_1, a_2, \dots, a_p)^\top$.

Then, the sample variance of $\mathbf{a}^\top \mathbf{X}$ can be represented by

$$V(\mathbf{a}^\top \mathbf{x}) = \mathbf{a}^\top \hat{\Sigma} \mathbf{a},$$

where $\hat{\Sigma} = V(\mathbf{X})$. $\mathbf{a}^\top \hat{\Sigma} \mathbf{a}$ is regarded as a p variable function of (a_1, a_2, \dots, a_p) : $\phi(a_1, a_2, \dots, a_p) = \mathbf{a}^\top \hat{\Sigma} \mathbf{a}$. To consider the optimization problem for ϕ , \mathbf{a} is constrained to $\mathbf{a}^\top \mathbf{a} = 1$. This problem is solved using Lagrange multipliers. The following Lagrange function is defined as

$$\begin{aligned} L(a_1, a_2, \dots, a_p) &= \phi(a_1, a_2, \dots, a_p) - \lambda_1 \left(\sum_{i=1}^p a_i^2 - 1 \right) \\ &= \mathbf{a}^\top \hat{\Sigma} \mathbf{a} - \lambda_1 (\mathbf{a}^\top \mathbf{a} - 1), \end{aligned}$$

where λ is the Lagrange multiplier. L is partially differentiated with respect to $\mathbf{a} = (a_1, a_2, \dots, a_p)^\top$ and λ_1 , and the derivatives are equated to zero. We therefore obtain the simultaneous equations:

$$\begin{cases} 2\hat{\Sigma} \mathbf{a} - 2\lambda_1 \mathbf{a} = 0 \\ \mathbf{a}^\top \mathbf{a} - 1 = 0. \end{cases}$$

This is an eigenvector problem; the solution to this problem for $\mathbf{a} = (a_1, a_2, \dots, a_p)^\top$ is a unit eigenvector of $\hat{\Sigma}$ corresponding to the largest eigenvalue. Let \mathbf{a} be an eigenvector and let λ be an eigenvalue. We then have

$$\phi(a_1, a_2, \dots, a_p) = V(\mathbf{a}^\top \mathbf{x}) = \mathbf{a}^\top \hat{\Sigma} \mathbf{a} = \lambda \mathbf{a}^\top \mathbf{a} = \lambda.$$

The eigenvector is denoted as \mathbf{a}_1 . Then $\mathbf{a}_1^\top \mathbf{x}_i; i = 1, 2, \dots, n$ are referred to as the first principal components. The first principal components are one-dimensional data that are the projection of the original data with the maximum variance. If all of the information for the data can be represented by the first principal components, further calculation is unnecessary. However, the first principal components usually exhibit the “size factor” only, whereas we would like to obtain another projection, namely the second principal components $\mathbf{a}_2^\top \mathbf{x}_i$.

The second principal components serve to explain the maximum variance under the constraint and the fact that they are independent of the first principal components. In other words, the second principal components $\mathbf{a}_2^\top \mathbf{X}$ take the maximum variance under the constraints $\mathbf{a}_1^\top \mathbf{a}_2 = 0$ and $\mathbf{a}_2^\top \mathbf{a}_2 = 1$. The second principal components can also be derived with Lagrange multipliers;

$$L(a_1, a_2, \dots, a_p, \lambda, \lambda_2) = \mathbf{a}^\top \hat{\Sigma} \mathbf{a} - \lambda \mathbf{a}_1^\top \mathbf{a} - \lambda_2 (\mathbf{a}^\top \mathbf{a} - 1).$$

L is partially differentiated with respect to $\mathbf{a} = (a_1, a_2, \dots, a_p)^\top$, λ and λ_2 , and the derivatives are equated to zero. The simultaneous equations below are obtained:

$$\begin{cases} 2\hat{\Sigma}\mathbf{a} - \lambda\mathbf{a}_1 - 2\lambda_2\mathbf{a} = 0 \\ \mathbf{a}_1^\top\mathbf{a} = 0 \\ \mathbf{a}_2^\top\mathbf{a} - 1 = 0. \end{cases}$$

We can obtain $\lambda = 0$ and λ_2 is another eigenvalue (not equal to λ_1). Since the variance of $\mathbf{a}_2^\top X$ is λ_2 , the \mathbf{a}_2 must be the second largest eigenvalue of $\hat{\Sigma}$. $\{\mathbf{a}_2^\top \mathbf{x}_i; i = 1, 2, \dots, n\}$ are referred to as the second principal components. The third principal components, fourth principal components, ..., and the p -th principal components can then be derived in the same manner.

Proportion and Accumulated Proportion

The first principal components through the p -th principal components were defined in the discussions above. As previously mentioned, the variance of the k -th principal components is λ_k . The sum of variances of p variables is $\sum_{j=1}^p \hat{\sigma}_j = \text{trace}(\hat{\Sigma})$, where $\hat{\Sigma} = (\hat{\sigma}_{ij})$. It is well known that $\text{trace}(\hat{\Sigma}) = \sum_{j=1}^p \lambda_j$; the sum of the variances coincides with the sum of the eigenvalues. The proportion of the k -th principal components is defined as the proportion of the entire variance to the variance of the k -th principal components:

$$\frac{\lambda_k}{\sum_{j=1}^p \lambda_j}.$$

The first principal components through the k -th principal components are generally used consecutively. The total variance of these principal components is represented by the accumulated proportion:

$$\frac{\sum_{j=1}^k \lambda_j}{\sum_{j=1}^p \lambda_j}.$$

We have explained PCA as an eigenvalue problem of covariance matrix. However, the results of this method are affected by units of measurements or scale transformations of variables. Thus, another method is to employ a correlation matrix rather than a covariance matrix. This method is invariant under units of variables, but does not take the variances of the variables into account.

6.2.2 Projection Pursuit

PCA searches a lower dimensional space that captures the majority of the variation within the data and discovers linear structures in the data. This method, however, is ineffective in analyzing nonlinear structures, i.e. curves, surfaces or clusters. In 1974, Friedman and Tukey (1974) proposed projection pursuit to search for linear projection onto the lower dimensional space that robustly reveals

structures in the data. After that, many researchers developed new methods for projection pursuit and evaluated them (e.g. Huber, 1985; Friedman, 1987; Hall, 1989; Iwasaki, 1991; Nason, 1995; Koyama et al., 1998). The fundamental idea behind projection pursuit is to search linear projection of the data onto a lower dimensional space their distribution is “interesting”; “interesting” is defined as being “far from the normal distribution”, i.e. the normal distribution is assumed to be the most uninteresting. The degree of “far from the normal distribution” is defined as being a projection index, the details of which will be described later.

Algorithm

The use of a projection index makes it possible to execute projection pursuit with the projection index. Here is the fundamental algorithm of k -dimensional projection pursuit.

1. Sphering \mathbf{x} : $\mathbf{z}_i = \hat{\Sigma}_{xx}^{-1/2}(\mathbf{x}_i - \hat{\mathbf{x}})$ ($i = 1, 2, \dots, n$), where $\hat{\Sigma}$ is the sample covariance matrix and $\hat{\mathbf{x}}$ is the sample mean of \mathbf{x} .
2. Initialize the project direction: $\boldsymbol{\alpha} = (\boldsymbol{\alpha}_1, \boldsymbol{\alpha}_2, \dots, \boldsymbol{\alpha}_k)$.
3. Search the direction $\boldsymbol{\alpha}$ that maximizes the projection index.
4. Project the data onto the lower dimensional space and display or analyze them.
5. Change the initial direction and repeat Steps 3 and 4, if necessary.

Projection Indexes

The goal of projection pursuit is to find a projection that reveals *interesting* structures in the data. There are various standards for interestingness, and it is a very difficult task to define. Thus, the normal distribution is regarded as uninteresting, and uninterestingness is defined as a degree that is “far from the normal distribution.”

Projection indexes are defined as of this degree. There are many definitions for projection indexes. Projection pursuit searches projections based on the projection index; methods of projection pursuit are defined by the projection indexes.

Here we will present several projection indexes. It is assumed that $\mathbf{Z} = (\mathbf{z}_1, \dots, \mathbf{z}_n)$ is the result of sphering \mathbf{X} ; the mean vector is a zero vector and the covariance matrix is an identity matrix.

Friedman’s Index. Friedman (1987) proposed the following projection index:

$$I = \frac{1}{2} \sum_{j=1}^J (2j+1) \left[\frac{1}{n} \sum_{i=1}^n P_j(2\Phi(\boldsymbol{\alpha}^\top \mathbf{Z}_i) - 1) \right]^2,$$

where $P_j(\cdot)$ are Legendre polynomials of order j and $\Phi(\cdot)$ is the cumulative distribution function of the normal distribution and J is a user-defined constant number, i.e. the degree of approximation.

In the case of two-dimensional projection pursuit, the index is represented by

$$\begin{aligned}
 I &= \sum_{j=1}^J (2j+1)E^2[P_j(R_1)]/4 \\
 &+ \sum_{k=1}^J (2k+1)E^2[P_k(R_2)]/4 \\
 &+ \sum_{j=1}^J \sum_{k=1}^{J-j} (2j+1)(2k+1)E^2[P_j(R_1)P_k(R_2)]/4,
 \end{aligned}$$

where

$$\begin{aligned}
 X_1 &= \boldsymbol{\alpha}_1^\top \mathbf{Z}, \quad X_2 = \boldsymbol{\alpha}_2^\top \mathbf{Z} \\
 R_1 &= 2\Phi(X_1) - 1, \quad R_2 = 2\Phi(X_2) - 1.
 \end{aligned}$$

Moment Index. The third and higher cumulants of the normal distribution vanish. The cumulants are sometimes used for the test of normality, i.e. they can be used for the projection index. Jones and Sibson (1987) proposed a one-dimensional projection index named the “moment index,” with the third cumulant $k_3 = \mu_3$ and the fourth cumulant $k_4 = \mu_4 - 3$:

$$I = k_3^2 + \frac{1}{4}k_4^2.$$

For two-dimensional projection pursuit, the moment index can be defined as

$$I = (k_{30}^2 + 3k_{21}^2 + 3k_{12}^2 + k_{03}^2) + \frac{1}{4}(k_{40}^2 + 4k_{31}^2 + 6k_{22}^2 + 4k_{13}^2 + k_{04}^2).$$

Hall’s Index. Hall (1989) proposed the following projection index:

$$I = \left[\theta_0(\boldsymbol{\alpha}) - 2^{-1/2} \pi^{-1/4} \right]^2 + \sum_{j=1}^J \theta_j^2(\boldsymbol{\alpha}),$$

where

$$\begin{aligned}
 \theta_j(\boldsymbol{\alpha}) &= n^{-1} \sum_{i=1}^n P_j(\boldsymbol{\alpha}^\top \mathbf{Z}_i) \phi(\boldsymbol{\alpha}^\top \mathbf{Z}_i), \\
 P_j(z) &= \frac{\sqrt{2}}{\sqrt{j!}} \pi^{1/4} H_j(2^{1/2}z),
 \end{aligned}$$

$\phi(z)$ is the normal density function and $H_j(z)$ are the Hermite polynomials of degree j . J is a user-defined constant number. Hall's index is much more robust for outliers than Freidman's index.

Relative Projection Pursuit

The main objective of ordinary projection pursuit is the discovery of non-normal structures in a dataset. Non-normality is evaluated using the degree of difference between the distribution of the projected dataset and the normal distribution.

There are times in which it is desired that special structures be discovered using criterion other than non-normal criterion. For example, if the purpose of analysis is to investigate a feature of a subset of the entire dataset, the projected direction should be searched so that the projected distribution of the subset is far from the distribution of the entire dataset. In sliced inverse regression (please refer to the final subsection of this chapter), the dataset is divided into several subsets based on the values of the response variable, and the effective dimension-reduction direction is searched for using projection pursuit. In this application of projection pursuit, projections for which the distributions of the projected subsets are far from those of the entire dataset are required. Mizuta (2002) proposed the adoption of relative projection pursuit for these purposes. Relative projection pursuit finds *interesting* low-dimensional space that differs from the reference dataset predefined by the user.

Nonlinear Reduction of High-dimensional Data

6.3

In the previous section, we discussed linear methods i.e. methods for dimension reduction through the use of linear projections. We will now move on to nonlinear methods for dimension reduction. First, we will describe a generalized principal component analysis (GPCA) method that is a nonlinear extension of PCA. Algebraic curve fitting methods will then be mentioned for a further extension of GPCA. Finally, we will introduce principal curves i.e. the parametric curves that pass through the middle of the data.

Generalized Principal Component Analysis

6.3.1

As long as data have a near-linear structure, the singularities of the data can be pointed out using PCA. On the contrary, if data have a nonlinear structure, GPCA will not be adequate for drawing conclusions regarding the nature of the data. To overcome this difficulty, GPCA has been proposed by Gnanadesikan and Wilk (1969), whereby fitting functions to the data points can be discovered.

Suppose that we have observations of p variables $\mathbf{x} = (x_1, x_2, \dots, x_p)$ on each of n individuals. Let $f_i(\mathbf{x}) (i = 1, 2, \dots, k)$ be k real-valued functions of the original variables.

The aim of GPCA is to discover a new set of variables (or functions of \mathbf{x}), as denoted by z_1, z_2, \dots, z_k , which are mutually uncorrelated and whose variances decrease, from first to last. Each z_j ($j = 1, 2, \dots, k$) is considered to be a linear combination of $f_i(\mathbf{x})$ ($i = 1, 2, \dots, k$), so that

$$z_j = \sum_{i=1}^k l_{ij} f_i(\mathbf{x}) = \mathbf{l}_j^\top \mathbf{f}(\mathbf{x}),$$

where $\mathbf{l}_j = (l_{1j}, l_{2j}, \dots, l_{kj})^\top$ are k constant vectors such that $\mathbf{l}_j^\top \mathbf{l}_j = 1$, and $\mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x}), \dots, f_k(\mathbf{x}))^\top$. The vectors $\mathbf{l}_1, \mathbf{l}_2, \dots, \mathbf{l}_k$ are the eigenvectors of the covariance matrix of $(f_1(\mathbf{x}), f_2(\mathbf{x}), \dots, f_k(\mathbf{x}))$, as in PCA. The function z_k defined by the “smallest” eigenvalue is considered to be one of the fitting functions to the data.

PCA is a special case of GPCA: real-valued functions $f_i(\mathbf{x})$ are reduced to x_i ($i = 1, 2, \dots, p$).

Quadratic principal component analysis (QPCA) is specified by the following functions:

$$\begin{cases} f_i(\mathbf{x}) = x_i & (i = 1, 2, \dots, p) \\ f_i(\mathbf{x}) = x_j x_m & (i = p + 1, \dots, (p^2 + 3p)/2), \end{cases}$$

where j, m is uniquely determined by

$$i = \{(2p - j + 3)j/2\} + m - 1, \\ 1 \leq j \leq m \leq p,$$

for $i(i = p + 1, \dots, (p^2 + 3p)/2)$.

QPCA for two dimensional data is defined by

$$\begin{aligned} f_1(x, y) &= x \\ f_2(x, y) &= y \\ f_3(x, y) &= x^2 \\ f_4(x, y) &= xy \\ f_5(x, y) &= y^2. \end{aligned}$$

Most GPCA methods are not invariant under orthogonal transformations and/or the translations (parallel transformations) of a coordinate system, though PCA is invariant under them. For example, QPCA is not invariant under them. The expression “the method is invariant” in this subsection means that the results of the method are never changed in the original coordinate by coordinate transformation. In the following, the determination of the GPCA methods that are invariant under the orthogonal transformations of a coordinate system will be described in the

case of two variables. Translations of a coordinate system are disregarded here because the data can be standardized to have a zero mean vector.

Hereafter, let us assume the following conditions:

A1 $f_1(\mathbf{x}), f_2(\mathbf{x}), \dots, f_k(\mathbf{x})$ are linearly independent as functions of \mathbf{x} .

A2 For any orthogonal matrix T , there is a matrix W such that $f(T\mathbf{x}) \equiv Wf(\mathbf{x})$.

A3 $f_i(\mathbf{x})$ are continuous functions.

Conditions A1 and A3 may be proper for GPCA, and condition A2 is necessary for discussing the influence of orthogonal coordinate transformations. PCA and QPCA clearly satisfy these conditions.

A GPCA method is referred to as “invariant” if its results in the original coordinate system are not changed by the orthogonal transformation of a coordinate system. It can be mathematically described as follows. For any orthogonal coordinate transformation: $\mathbf{x}^* = T\mathbf{x}$,

$$\begin{aligned} z_j^* &= \mathbf{l}_j^{*\top} \mathbf{f}(\mathbf{x}^*) \\ &= \mathbf{l}_j^{*\top} \mathbf{f}(T\mathbf{x}) \quad (j = 1, 2, \dots, k) \end{aligned}$$

denote the results of the method for transformed variables \mathbf{x}^* , where \mathbf{l}_j^* are eigenvectors of $\text{Cov}(\mathbf{f}(\mathbf{x}^*))$. The method is “invariant” if it holds that

$$\mathbf{l}_j^\top \mathbf{f}(\mathbf{x}) \equiv \pm \mathbf{l}_j^{*\top} \mathbf{f}(T\mathbf{x}) \quad (j = 1, 2, \dots, k)$$

as vector-valued functions of \mathbf{x} for any orthogonal matrix T . The plus or minus sign is indicated only for the orientations of the eigenvectors.

The GPCA method specified by $\mathbf{f}(\mathbf{x})$ is invariant under an orthogonal transformation, if and only if the matrix W is an orthogonal matrix for any orthogonal matrix T . The proof will be described below. If the method is invariant, W can be taken as

$$(\mathbf{l}_1^*, \mathbf{l}_2^*, \dots, \mathbf{l}_k^*) (\mathbf{l}_1, \mathbf{l}_2, \dots, \mathbf{l}_k)^\top,$$

which is an orthogonal matrix. Conversely, if W is an orthogonal matrix, $W^\top \mathbf{l}_j^*$ are eigenvectors of $\text{Cov}(\mathbf{f}(\mathbf{x}))$. Therefore the following is obtained:

$$\mathbf{l}_j^\top = \pm \mathbf{l}_j^{*\top} W.$$

Mizuta (1983) derived a theorem on invariant GPCA.

Theorem 1 GPCA methods for two-dimensional data (x, y) under the conditions A1, A2 and A3 that are invariant under rotations can be restricted to those specified by the following functions.

(1) s pairs of functions:

$$\left\{ \begin{array}{l} f_{2i-1}(x, y) = g_i(\sqrt{x^2 + y^2}) \left(x^{N_i} - \binom{N_i}{2} y^2 x^{N_i-2} + \binom{N_i}{4} y^4 x^{N_i-4} - \dots \right) \\ \quad - h_i(\sqrt{x^2 + y^2}) \left(N_i y x^{N_i-1} - \binom{N_i}{3} y^3 x^{N_i-3} + \binom{N_i}{5} y^5 x^{N_i-5} - \dots \right) \\ f_{2i}(x, y) = g_i(\sqrt{x^2 + y^2}) \left(N_i y x^{N_i-1} - \binom{N_i}{3} y^3 x^{N_i-3} + \binom{N_i}{5} y^5 x^{N_i-5} - \dots \right) \\ \quad + h_i(\sqrt{x^2 + y^2}) \left(x^{N_i} - \binom{N_i}{2} y^2 x^{N_i-2} + \binom{N_i}{4} y^4 x^{N_i-4} - \dots \right) \end{array} \right.$$

$(i = 1, 2, \dots, s),$

where g_i, h_i are arbitrary continuous functions of $\sqrt{x^2 + y^2}$ and N_i are arbitrary positive integers.

(2) Continuous functions of $\sqrt{x^2 + y^2}$.

The above theorem can be extended for use with GPCA methods for p -dimensional data because invariant GPCA for p -dimensional data methods are invariant under the rotations of any pair of two variables and the reverse is also true.

We will show some set of functions for invariant GPCA here.

(1) 3 dimensional and degree 1:

$$x, y, z.$$

(2) 3 dimensional and degree 2:

$$x^2, y^2, z^2, \sqrt{2}xy, \sqrt{2}yz, \sqrt{2}zx.$$

(3) 3 dimensional and degree 3:

$$x^3, y^3, z^3, \sqrt{3}x^2y, \sqrt{3}y^2z, \sqrt{3}z^2x, \sqrt{3}xy^2, \sqrt{3}yz^2, \sqrt{3}zx^2, \sqrt{6}xyz.$$

(4) 3 dimensional and degree q :

$$\sqrt{\frac{q!}{i!j!k!}} x^i y^j z^k$$

$$(i + j + k = q; \quad 0 \leq i, j, k).$$

(5) p dimensional and degree q :

$$\sqrt{\frac{q!}{\prod_{i=1}^p k_i!}} \prod_{t=1}^p (x_t)^{k_t}$$

$$\sum_{t=1}^p k_t = q; \quad 0 \leq k_t .$$

Algebraic Curve and Surface Fitting

6.3.2

Next, we will discuss a method involving algebraic curve and surface fitting to multidimensional data.

The principal component line minimizes the sum of squared deviations in each of the variables. The PCA cannot find non-linear structures in the data. GPCA is used to discover an algebraic curve fitted to data; the function z_k defined by the “smallest” eigenvalue is considered to be one of the fitting functions to the data. However, it is difficult to interpret algebraic curves statistically derived from GPCA.

We will now describe methods for estimating the algebraic curve or surface that minimizes the sum of squares of perpendicular distances from multidimensional data.

Taubin (1991) developed an algorithm for discovering the algebraic curve for which the sum of *approximate* squares distances between data points and the curve is minimized. The approximate squares distance does not always agree with the *exact* squares distance. Mizuta (1995, 1996) presented an algorithm for evaluating the exact distance between the data point and the curve, and have presented a method for algebraic curve fitting with exact distances. In this subsection, we describe the method of algebraic surface fitting with exact distances. The method of the algebraic curve fitting is nearly identical to that of surface fitting, and is therefore omitted here.

Algebraic Curve and Surface

A p -dimensional algebraic curve or surface is the set of zeros of k -polynomials $\mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), \dots, f_k(\mathbf{x}))$ on \mathbb{R}^p ,

$$Z(\mathbf{f}) = \{\mathbf{x} : \mathbf{f}(\mathbf{x}) = 0\} .$$

In the case of $p = 2$ and $k = 1$, $Z(\mathbf{f})$ is a curve in the plane. For example, $Z(x^2 + 2y^2 - 1)$ is an ellipse and $Z(y^2 - x^2 + 1)$ is a hyperbola. In the case of $p = 3$ and $k = 2$, $Z(\mathbf{f})$ is a curve in the space.

In the case of $p = 3$ and $k = 1$, $Z(\mathbf{f})$ is a surface:

$$Z(\mathbf{f}) = \{(x, y, z) : f(x, y, z) = 0\} .$$

Hereafter, we will primarily discuss this case.

Approximate Distance

The distance from a point \mathbf{a} to the surface $Z(f)$ is usually defined by

$$\text{dist}(\mathbf{a}, Z(f)) = \inf (\| \mathbf{a} - \mathbf{y} \| : \mathbf{y} \in Z(f)) .$$

It was said that the distance between a point and the algebraic curve or surface cannot be computed using direct methods. Thus, Taubin proposed *an approximate distance* from \mathbf{a} to $Z(f)$ (Taubin, 1991). The point $\hat{\mathbf{y}}$ that approximately minimizes the distance $\| \mathbf{y} - \mathbf{a} \|$, is given by

$$\hat{\mathbf{y}} = \mathbf{a} - (\nabla f(\mathbf{a})^\top)^\dagger f(\mathbf{a}) ,$$

where $(\nabla f(\mathbf{a})^\top)^\dagger$ is the pseudoinverse of $\nabla f(\mathbf{a})^\top$. The distance from \mathbf{a} to $Z(f)$ is approximated to

$$\text{dist}(\mathbf{a}, Z(f))^2 \approx \frac{f(\mathbf{a})^2}{\| \nabla f(\mathbf{a}) \|^2} .$$

Taubin also presented an algorithm to find the algebraic curve for which the sum of *approximate* squares distances between data points and the curve is minimized.

Exact Distance

In the following, we present a method for calculating the distance between a point $\mathbf{a} = (\alpha, \beta, \gamma)$ and an algebraic surface $Z(f)$.

If (x, y, z) is the nearest point to the point $\mathbf{a} = (\alpha, \beta, \gamma)$ on $Z(f)$, (x, y, z) satisfies the following simultaneous equations:

$$\begin{cases} \phi_1(x, y, z) = 0 \\ \phi_2(x, y, z) = 0 \\ f(x, y, z) = 0 , \end{cases} \quad (6.1)$$

where $\phi_1(x, y, z) = (x - \alpha)(\partial f / \partial y) - (y - \beta)(\partial f / \partial x)$, and $\phi_2(x, y, z) = (z - \gamma)(\partial f / \partial y) - (y - \beta)(\partial f / \partial z)$.

Equations (6.1) can be solved using the Newton-Rapson method:

1. Set x_0, y_0 and z_0 (see below).
2. Solve the equations:

$$\begin{cases} h \frac{\partial \phi_1}{\partial x} + k \frac{\partial \phi_1}{\partial y} + l \frac{\partial \phi_1}{\partial z} = -\phi_1(x, y, z) \\ h \frac{\partial \phi_2}{\partial x} + k \frac{\partial \phi_2}{\partial y} + l \frac{\partial \phi_2}{\partial z} = -\phi_2(x, y, z) \\ h \frac{\partial f}{\partial x} + k \frac{\partial f}{\partial y} + l \frac{\partial f}{\partial z} = -f(x, y, z) . \end{cases} \quad (6.2)$$

3. Replace x, y :

$$\begin{cases} x_{i+1} = x_i + h \\ y_{i+1} = y_i + k \\ z_{i+1} = z_i + l. \end{cases}$$

4. Stop if $h^2 + k^2 + l^2$ is below a certain threshold. Otherwise, go to Step 2.

One of the important points to consider when applying the Newton–Raphson method is to compute an initial point. We have a good initial point: (α, β, γ) .

When $x_0 = \alpha, y_0 = \beta, z_0 = \gamma$, (6.2) are

$$\begin{cases} h \frac{\partial \phi_1}{\partial x} + k \frac{\partial \phi_1}{\partial y} + l \frac{\partial \phi_1}{\partial z} = 0 \\ h \frac{\partial \phi_2}{\partial x} + k \frac{\partial \phi_2}{\partial y} + l \frac{\partial \phi_2}{\partial z} = 0 \\ h \frac{\partial f}{\partial x} + k \frac{\partial f}{\partial y} + l \frac{\partial f}{\partial z} = -f(x, y, z). \end{cases}$$

It is very simple to show that the distance between (x_1, y_1, z_1) and (α, β, γ) agrees with Taubin's approximate distance.

Algebraic Surface Fitting

We have already described the method for calculating the distance between a point and a surface.

The problem of finding a fitting surface that minimizes the sum of the distances from data points can therefore be solved by using an optimization method without derivatives. However, for computing efficiency, the partial derivatives of the sum of squares of distances from data with the coefficients of an algebraic curve are derived.

In general, a polynomial f in a set is denoted by

$$f(b_1, \dots, b_q; x, y, z),$$

where b_1, \dots, b_q are the parameters of the set.

Let $\mathbf{a}_i = (\alpha_i, \beta_i, \gamma_i)$ ($i = 1, 2, \dots, n$) be n data points within the space. The point in $Z(f)$ that minimizes the distance from $(\alpha_i, \beta_i, \gamma_i)$ is denoted by (x_i, y_i, z_i) ($i = 1, 2, \dots, n$).

The sum of squares of distances is

$$R = \sum_{i=1}^n (\mathbf{x}_i - \mathbf{a}_i)^\top (\mathbf{x}_i - \mathbf{a}_i).$$

R can be minimized with respect to the parameters of polynomial f with the *Levenberg–Marquardt Method*. This method requires partial derivatives of R with respect to b_j :

$$\frac{\partial R}{\partial b_j} = \sum_{i=1}^n \frac{\partial R_i}{\partial b_j}, \quad (6.3)$$

where

$$\frac{\partial R_i}{\partial b_j} = 2 \left((x_i - \alpha_i) \frac{\partial x_i}{\partial b_j} + (y_i - \beta_i) \frac{\partial y_i}{\partial b_j} + (z_i - \gamma_i) \frac{\partial z_i}{\partial b_j} \right). \quad (6.4)$$

The only matter left to discuss is a solution for $\partial x_i / \partial b_j$, $\partial y_i / \partial b_j$ and $\partial z_i / \partial b_j$. Hereafter, the subscript i is omitted. By the derivative of both sides of $f(b_1, \dots, b_q, x, y, z) = 0$ with respect to b_j ($j = 1, \dots, q$), we obtain

$$\frac{\partial f}{\partial x} \frac{\partial x}{\partial b_j} + \frac{\partial f}{\partial y} \frac{\partial y}{\partial b_j} + \frac{\partial f}{\partial z} \frac{\partial z}{\partial b_j} + \frac{df}{db_j} = 0, \quad (6.5)$$

where df/db_j is the differential of f with b_j when x and y are fixed.

Since \mathbf{x}_i is on the normal line from \mathbf{a}_i ,

$$\left(\left. \frac{\partial f}{\partial x} \right|_{\mathbf{x}_i}, \left. \frac{\partial f}{\partial y} \right|_{\mathbf{x}_i}, \left. \frac{\partial f}{\partial z} \right|_{\mathbf{x}_i} \right)^\top (\mathbf{x}_i - \mathbf{a}_i) = 0.$$

By the derivative of

$$(y - \beta)(z - \gamma) \left. \frac{\partial f}{\partial x} \right|_x = t$$

$$(x - \alpha)(z - \gamma) \left. \frac{\partial f}{\partial y} \right|_x = t$$

$$(x - \alpha)(y - \beta) \left. \frac{\partial f}{\partial z} \right|_x = t$$

with respect to b_j , we obtain the linear combinations of $\partial x / \partial b_j$, $\partial y / \partial b_j$ and $\partial z / \partial b_j$:

$$c_{1m} \frac{\partial x}{\partial b_j} + c_{2m} \frac{\partial y}{\partial b_j} + c_{3m} \frac{\partial z}{\partial b_j} + c_{4m} = \frac{\partial t}{\partial b_j}, \quad (6.6)$$

where c_{1m}, \dots, c_{4m} are constants ($m = 1, \dots, 3$).

Equations (6.5) and (6.6) are simultaneous linear equations in four variables $\partial x / \partial b_j$, $\partial y / \partial b_j$, $\partial z / \partial b_j$ and $\partial t / \partial b_j$. We then obtain $\partial x / \partial b_j$, $\partial y / \partial b_j$ and $\partial z / \partial b_j$ at (x_i, y_i, z_i) . By (6.4), we have the partial differentiation of R_i with respect to b_j .

Therefore, we can obtain the algebraic curve that minimizes the sum of squares of distances from data points with the Levenberg–Marquardt method.

Bounded and Stably Bounded Algebraic Curve and Surface

Although algebraic curves can fit the data very well, they usually contain points far remote from the given data set. In 1994, Keren et al. (1994) and Taubin et al. (1994) independently developed algorithms for a *bounded* (closed) algebraic curve with approximate squares distance. We will now introduce the definition and properties of a bounded algebraic curve.

$Z(f)$ is referred to as *bounded* if there exists a constant r such that $Z(f) \subset \{\mathbf{x} : \|\mathbf{x}\| < r\}$. For example, it is clear that $Z(x^2 + y^2 - 1)$ is bounded, but $Z(x^2 - y^2)$ is not bounded.

Keren et al. (1994) defined $Z(f)$ to be *stably bounded* if a small perturbation of the coefficients of the polynomial leaves its zero set bounded. An algebraic curve $Z((x - y)^4 + x^2 + y^2 - 1)$ is bounded but not stably bounded because $Z((x - y)^4 + x^2 + y^2 - 1 + \epsilon x^3)$ is not bounded for any $\epsilon \neq 0$.

Let $f_k(x, y)$ be the form of degree k of a polynomial $f(x, y): f(x, y) = \sum_{k=0}^d f_k(x, y)$. The leading form of a polynomial $f(x, y)$ of degree d is defined by $f_d(x, y)$. For example, the leading form of $f(x, y) = x^2 + 2xy - y^2 + 5x - y + 3$ is $f_2(x, y) = x^2 + 2xy - y^2$.

Lemma 1 For an even positive integer d , any leading form $f_d(x, y)$ can be represented by $\mathbf{x}^\top A \mathbf{x}$. Where A is a symmetric matrix and $\mathbf{x} = (x^{d/2}, x^{d/2-1}y, \dots, xy^{d/2-1}, y^{d/2})^\top$. 1

Theorem 2 (Keren et al., 1994): The $Z(f)$ is stably bounded if and only if d is even and there exists a symmetric positive definite matrix A such that 2

$$f_d(x, y) = \mathbf{x}^\top A \mathbf{x},$$

where $\mathbf{x} = (x^{d/2}, x^{d/2-1}y, \dots, xy^{d/2-1}, y^{d/2})^\top$.

These definitions and theorem for algebraic curves are valid for algebraic surfaces. Hereafter, we will restrict our discussion to algebraic *surfaces*.

Parameterization

We parameterize the set of all polynomials of degree k and the set of polynomials that induce (stably) bounded algebraic surfaces. In general, a polynomial f of degree p with q parameters can be denoted by $f(b_1, \dots, b_q; x, y)$, where b_1, \dots, b_q are the parameters of the polynomial.

For example, *all of the polynomials* of degree 2 can be represented by

$$f(b_1, b_2, \dots, b_{10}; x, y, z) = B^\top X,$$

where $X = (1, x, y, z, x^2, y^2, z^2, xy, yz, zx)^\top$, $B = (b_1, b_2, \dots, b_{10})^\top$.

For stably bounded algebraic curves of degree 4,

$$f(b_1, \dots, b_{41}; x, y, z) \\ = (x^2, y^2, z^2, xy, yz, zx) A^2 (x^2, y^2, z^2, xy, yz, zx)^\top + (b_{22}, \dots, b_{41}) (1, x, y, z, \dots, z^3)^\top,$$

where

$$A = \begin{pmatrix} b_1 & b_2 & b_3 & b_4 & b_5 & b_6 \\ b_2 & b_7 & b_8 & b_9 & b_{10} & b_{11} \\ b_3 & b_8 & b_{12} & b_{13} & b_{14} & b_{15} \\ b_4 & b_9 & b_{13} & b_{16} & b_{17} & b_{18} \\ b_5 & b_{10} & b_{14} & b_{17} & b_{19} & b_{20} \\ b_6 & b_{11} & b_{15} & b_{18} & b_{20} & b_{21} \end{pmatrix}.$$

Examples

Here we will show a numerical example of the algebraic surface and bounded algebraic surface fitting methods.

The data in this example is three-dimensional data of size 210. The 210 points nearly lie on a closed cylinder (Fig. 6.1). The result of GPCA is set for an initial surface and the method is used to search for a fitting algebraic surface of degree 4 (Figs. 6.2, 6.3 and 6.4). The value of R is 0.924.

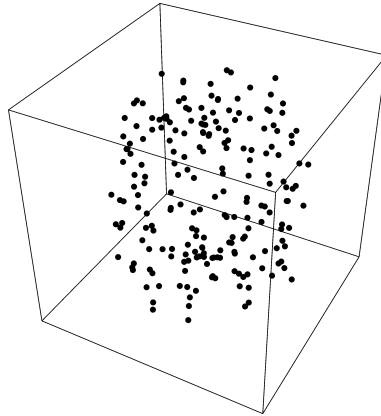


Figure 6.1. Surface fitting for distributed cylinder data (Original Data Points)

Figure 6.5 presents the result of a *bounded* algebraic surface fitting the same data. The value of R is 1.239, and is greater than that of unbounded fitting. The bounded surface, however, directly reveals the outline of the data.

In this subsection, we have discussed algebraic surface fitting to multidimensional data. Two sets of algebraic surfaces were described: an unbounded algebraic

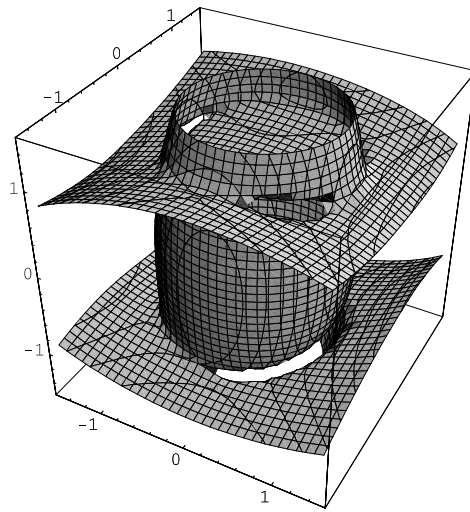


Figure 6.2. Surface fitting for distributed cylinder data (Unbounded Fitting Surface)

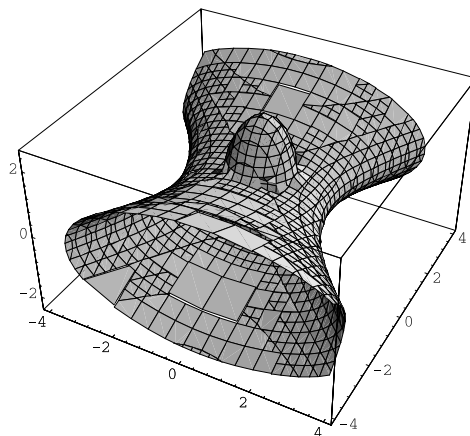


Figure 6.3. Surface fitting for distributed cylinder data (Global View of 2)

surface and a bounded algebraic surface. This method can be extended for use with any other family of algebraic surfaces.

Taubin (1994) proposed the approximate distance of order k and presented algorithms for rasterizing algebraic curves. The proposed algorithm for exact distance can also be used for rasterizing algebraic curves and surfaces. Mizuta (1997) has successfully developed a program for rasterizing them with exact distances.

Principal Curves

6.3.3

Curve fitting to data is an important method for data analysis. When we obtain a fitting curve for data, the dimension of the data is nonlinearly reduced to one di-

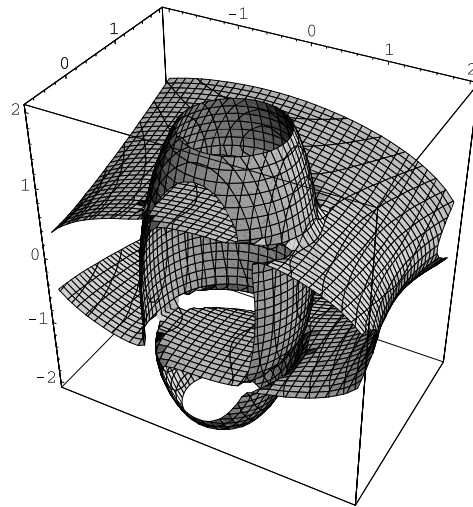


Figure 6.4. Surface fitting for distributed cylinder data (Cutting View of 2)

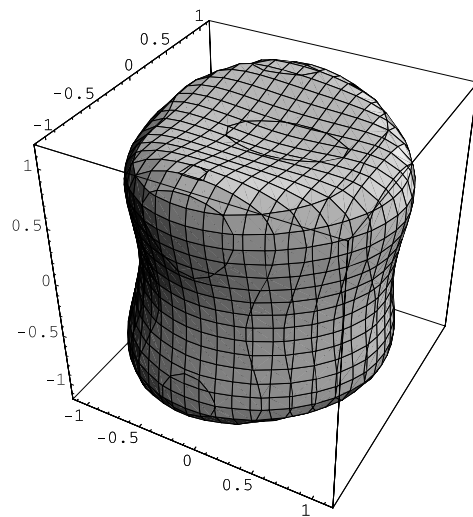


Figure 6.5. Surface fitting for distributed cylinder data (Bounded Fitting Surface)

mension. Hastie and Stuetzle (1989) proposed the concept of a principal curve and developed a concrete algorithm to find the principal curve, which is represented by a parametric curve. We can therefore obtain a new nonlinear coordinate for the data using the principal curve.

Definition of Principal Curve

First, we will define principal curves for a p -dimensional distribution function $h(\mathbf{x})(\mathbf{x} \in R^p)$, rather than a dataset.

The expectation of X with density function h in R^p is denoted by $E_h(X)$. The parametric curve within the p -dimensional space is represented by $f(\lambda)$, where λ is the parameter.

For each point \mathbf{x} in R^p , the parameter λ of the nearest point on the curve $f(\lambda)$ is denoted by $\lambda_f(\mathbf{x})$, which is referred to as the *projection index*. The projection index, which is different from projection index in projection pursuit, is defined as follows:

$$\lambda_f(\mathbf{x}) = \sup_{\lambda} \left\{ \lambda \mid \|\mathbf{x} - f(\lambda)\| = \inf_{\mu} \|\mathbf{x} - f(\mu)\| \right\} .$$

The curve $f(\lambda)$ is referred to as the principal curve of density function h , if

$$E_h(\mathbf{x} \mid \lambda_f(\mathbf{x}) = \lambda) = f(\lambda) \quad (\text{for a.e. } \lambda)$$

is satisfied. After all, for any point $f(\lambda)$ on the curve, the average of the conditional distribution of \mathbf{x} given $\lambda_f(\mathbf{x}) = \lambda$ is consistent with $f(\lambda)$ with the exception of a set of measure 0.

The principal curves of a given distribution are not always unique. For example, two principal components of the two-dimensional normal distribution are principal curves.

The algorithm for finding the principal curves of a distribution is:

1. *Initialization.* Put

$$f^{(0)}(\lambda) = \bar{\mathbf{x}} + \mathbf{a}\lambda ,$$

where \mathbf{a} is the first principal component of the distribution defined by the density function h and $\bar{\mathbf{x}}$ is the average of \mathbf{x} .

2. *Expectation Step* (update of $f(\lambda)$).

$$f^{(j)}(\lambda) = E(\mathbf{x} \mid \lambda_{f^{(j-1)}}(\mathbf{x}) = \lambda) \quad \forall \lambda$$

3. *Projection Step* (update of λ).

$$\lambda^{(j)}(\mathbf{x}) = \lambda_{f^{(j)}}(\mathbf{x}) \quad \forall \mathbf{x} \in R^p$$

And transform the $\lambda^{(j)}$ to be arc length.

4. *Evaluation.* Calculate

$$D^2(h, f^{(j)}) = E_{\lambda^{(j)}} E \left\{ \|\mathbf{x} - f(\lambda^{(j)}(\mathbf{x}))\|^2 \mid \lambda^{(j)}(\mathbf{x}) \right\} .$$

If the value

$$\frac{|D^2(h, f^{(j-1)}) - D^2(h, f^{(j)})|}{D^2(h, f^{(j-1)})}$$

is smaller than ε , then stop, otherwise $j = j + 1$ and go to Step 1.

In the Expectation Step, calculate the expectation with respect to the distribution h of the set of \mathbf{x} satisfying $\lambda_{f^{(j-1)}}(\mathbf{x}) = \lambda$ and substitute $f^{(j)}(\lambda)$ for it. In the Projection Step, project data points in R^p to the curve $f^{(j)}(\lambda)$ and assign $\lambda^{(j)}(\mathbf{x})$.

For actual data analysis, only a set of data points is given and the distribution is unknown. Hastie and Stuetzle (1989) also proposed an algorithm with which to derive the principal curve for given p -dimensional data of size n : x_{ik} ($i = 1, 2, \dots, N; k = 1, 2, \dots, p$). In this case, the principal curves are represented by lines determined by N points (λ_i, f_i) .

1. *Initialization.*

$$f^{(0)}(\lambda) = \bar{\mathbf{x}} + \mathbf{u}\lambda,$$

where \mathbf{u} is the first principal component of the data and $\bar{\mathbf{x}}$ is the average of \mathbf{x} .

2. *Expectation Step.* Smooth x_{ik} ($i = 1, 2, \dots, N$) with respect to λ for each k independently and calculate $f^{(j)}(\lambda)$.
3. *Projection Step.* Search for the nearest point on the curve (line curve) of each data point and assign it to their value of λ .
4. *Evaluation.* If a terminal condition is satisfied, the algorithm is stopped. If not, $j = j + 1$ and go to Step 2.

Linear Reduction of Explanatory Variables

6.4

Thus far, we have described dimension reduction methods for multidimensional data, where there are no distinctions among variables. However, there are times when we must analyze multidimensional data in which a variable is a response variable and others are explanatory variables. Regression analysis is usually used for the data. Dimension reduction methods of explanatory variables are introduced below.

Sliced Inverse Regression

Regression analysis is one of the fundamental methods used for data analysis. A response variable y is estimated by a function of explanatory variables \mathbf{x} , a p -dimensional vector. An immediate goal of ordinary regression analysis is to find the function of \mathbf{x} . When there are many explanatory variables in the data set, it is difficult to stably calculate the regression coefficients. An approach to reducing the number of explanatory variables is explanatory variable selection, and there are many studies on variable selection. Another approach is to project the explanatory variables on a lower dimensional space that nearly estimates the response variable.

Sliced Inverse Regression (SIR), which was proposed by Li (1991), is a method that can be employed to reduce explanatory variables with linear projection. SIR finds linear combinations of explanatory variables that are a reduction for non-linear regression. The original SIR algorithm, however, cannot derive suitable

results for some artificial data with trivial structures. Li also developed another algorithm, SIR2, which uses the conditional estimation $E[\text{cov}(\mathbf{x}|y)]$. However, SIR2 is also incapable of finding trivial structures for another type of data.

We hope that projection pursuit can be used for finding linear combinations of explanatory variables. A new SIR method with projection pursuit (SIRpp) is described here. We also present a numerical example of the proposed method.

Sliced Inverse Regression Model

SIR is based on the model (SIR model):

$$y = f(\boldsymbol{\beta}_1^\top \mathbf{x}, \boldsymbol{\beta}_2^\top \mathbf{x}, \dots, \boldsymbol{\beta}_K^\top \mathbf{x}) + \varepsilon, \quad (6.7)$$

where \mathbf{x} is the vector of p explanatory variables, $\boldsymbol{\beta}_k$ are unknown vectors, ε is independent of \mathbf{x} , and f is an arbitrary unknown function on \mathbf{R}^K .

The purpose of SIR is to estimate the vectors $\boldsymbol{\beta}_k$ for which this model holds. If we obtain $\boldsymbol{\beta}_k$, we can reduce the dimension of \mathbf{x} to K . Hereafter, we shall refer to any linear combination of $\boldsymbol{\beta}_k$ as the effective dimensional reduction (e.d.r.) direction.

Li (1991) proposed an algorithm for finding e.d.r. directions, and it was named SIR. However, we refer to the algorithm as SIR1 to distinguish it from the SIR model.

The main idea of SIR1 is to use $E[\mathbf{x}|y]$. $E[\mathbf{x}|y]$ is contained in the space spanned by e.d.r. directions, but there is no guarantee that $E[\mathbf{x}|y]$ will span the space. For example, in Li, if $(X_1, X_2) \sim N(0, I_2)$, $Y = X_1^2$ then $E[X_1|y] = E[X_2|y] = 0$.

SIR Model and Non-Normality

Hereafter, it is assumed that the distribution of \mathbf{x} is standard normal distribution: $\mathbf{x} \sim N(0, I_p)$. If not, standardize \mathbf{x} by affine transformation. In addition, $\boldsymbol{\beta}_i^\top \boldsymbol{\beta}_j = \delta_{ij}$, ($i, j = 1, 2, \dots, K$) is presumed without loss of generality. We can choose $\boldsymbol{\beta}_i$ ($i = K + 1, \dots, p$) such that $\{\boldsymbol{\beta}_i\}$ ($i = 1, \dots, p$) is a basis for \mathbf{R}^p .

Since the distribution of \mathbf{x} is $N(0, I_p)$, the distribution of $(\boldsymbol{\beta}_1^\top \mathbf{x}, \dots, \boldsymbol{\beta}_p^\top \mathbf{x})$ is also $N(0, I_p)$. The density function of $(\boldsymbol{\beta}_1^\top \mathbf{x}, \dots, \boldsymbol{\beta}_p^\top \mathbf{x}, y)$ is

$$\begin{aligned} & h(\boldsymbol{\beta}_1^\top \mathbf{x}, \dots, \boldsymbol{\beta}_p^\top \mathbf{x}, y) \\ &= \phi(\boldsymbol{\beta}_1^\top \mathbf{x}) \dots \phi(\boldsymbol{\beta}_p^\top \mathbf{x}) \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y - f(\boldsymbol{\beta}_1^\top \mathbf{x}, \dots, \boldsymbol{\beta}_K^\top \mathbf{x}))^2}{2\sigma^2}\right), \end{aligned}$$

where $\phi(\mathbf{x}) = 1/\sqrt{2\pi} \exp(-x^2/2)$ and we assume $\varepsilon \sim N(0, \sigma^2)$.

The conditional density function is

$$h(\boldsymbol{\beta}_1^\top \mathbf{x}, \dots, \boldsymbol{\beta}_p^\top \mathbf{x} | y) = \phi(\boldsymbol{\beta}_{K+1}^\top \mathbf{x}) \dots \phi(\boldsymbol{\beta}_p^\top \mathbf{x}) g(\boldsymbol{\beta}_1^\top \mathbf{x}, \dots, \boldsymbol{\beta}_K^\top \mathbf{x}),$$

where $g()$ is a function of $\boldsymbol{\beta}_1^\top \mathbf{x}, \dots, \boldsymbol{\beta}_K^\top \mathbf{x}$, which is not generally the normal density function.

Thus, $h(\boldsymbol{\beta}_1^\top \mathbf{x}, \dots, \boldsymbol{\beta}_p^\top \mathbf{x} \mid y)$ is separated into the normal distribution part $\phi(\boldsymbol{\beta}_{K+1}^\top \mathbf{x}) \dots \phi(\boldsymbol{\beta}_p^\top \mathbf{x})$ and the non-normal distribution part $g(\cdot)$.

Projection Pursuit is an excellent method for finding non-normal parts, so we adopt it for SIR.

SIRpp Algorithm

Here we show the algorithm for the SIR model with projection pursuit (SIRpp). The algorithm for the data (y_i, \mathbf{x}_i) ($i = 1, 2, \dots, n$) is as follows:

1. Standardize \mathbf{x} : $\tilde{\mathbf{x}}_i = \hat{\Sigma}_{\mathbf{xx}}^{-1/2}(\mathbf{x}_i - \bar{\mathbf{x}})$ ($i = 1, 2, \dots, n$), where $\hat{\Sigma}_{\mathbf{xx}}$ is the sample covariance matrix and $\bar{\mathbf{x}}$ is the sample mean of \mathbf{x} .
2. Divide the range of y into H slices, I_1, \dots, I_H .
3. Conduct a projection pursuit in K dimensional space for each slice. The following H projections are obtained: $(\boldsymbol{\alpha}_1^{(h)}, \dots, \boldsymbol{\alpha}_K^{(h)})$, ($h = 1, \dots, H$).
4. Let the K largest eigenvectors of \hat{V} be $\hat{\boldsymbol{\eta}}_k$ ($k = 1, \dots, K$). Output $\hat{\boldsymbol{\beta}}_k = \hat{\boldsymbol{\eta}}_k \hat{\Sigma}_{\mathbf{xx}}^{-1/2}$ ($k = 1, 2, \dots, K$) for the estimation of e.d.r. directions, where $\hat{V} = \sum_{h=1}^H w(h) \sum_{k=1}^K \boldsymbol{\alpha}_k^{(h)\top} \boldsymbol{\alpha}_k^{(h)}$.

Numerical Examples

Two models of the multicomponent are used:

$$y = x_1(x_1 + x_2 + 1) + \sigma \cdot \varepsilon, \quad (6.8)$$

$$y = \sin(x_1) + \cos(x_2) + \sigma \cdot \varepsilon \quad (6.9)$$

to generate $n = 400$ data, where $\sigma = 0.5$. We first generate x_1, x_2, ε with $N(0, 1)$ and calculate response variable y using (6.8) or (6.9). Eight variables x_3, \dots, x_{10} generated by $N(0, 1)$ are added to the explanatory variables. The ideal e.d.r. directions are contained within the space spanned by two vectors $(1, 0, \dots, 0)$ and $(0, 1, \dots, 0)$.

The squared multiple correlation coefficient between the projected variable $\mathbf{b}^\top \mathbf{x}$ and the space B spanned by ideal e.d.r. directions:

$$R^2(\mathbf{b}) = \max_{\boldsymbol{\beta} \in B} \frac{(\mathbf{b}^\top \sum_{\mathbf{xx}} \boldsymbol{\beta})^2}{\mathbf{b}^\top \sum_{\mathbf{xx}} \mathbf{b} \cdot \boldsymbol{\beta}^\top \sum_{\mathbf{xx}} \boldsymbol{\beta}} \quad (6.10)$$

is adopted as the criterion for evaluating the effectiveness of estimated e.d.r. directions.

Table 6.1 shows the mean and the standard deviation (in parentheses) of $R^2(\hat{\boldsymbol{\beta}}_1)$ and $R^2(\hat{\boldsymbol{\beta}}_2)$ of four SIR algorithms for $H = 5, 10$, and 20 , after 100 replicates. SIR2 cannot reduce the explanatory variables from the first example. The result of the second example is very interesting. SIR1 finds the asymmetric e.d.r. direction, but, does not find the symmetric e.d.r. direction. Conversely, SIR2 finds only the symmetric e.d.r. direction. SIRpp can detect both of the e.d.r. directions.

Table 6.1. Results for SIR1, SIR2, and SIRpp (Example 1)

H	SIR1		SIR2		SIRpp	
	$R^2(\hat{\beta}_1)$	$R^2(\hat{\beta}_2)$	$R^2(\hat{\beta}_1)$	$R^2(\hat{\beta}_2)$	$R^2(\hat{\beta}_1)$	$R^2(\hat{\beta}_2)$
5	0.92 (0.04)	0.77 (0.11)	0.96 (0.03)	0.20 (0.21)	0.97 (0.02)	0.78 (0.15)
10	0.93 (0.03)	0.81 (0.09)	0.92 (0.09)	0.10 (0.12)	0.95 (0.04)	0.79 (0.13)
20	0.92 (0.04)	0.76 (0.18)	0.83 (0.19)	0.11 (0.13)	0.95 (0.07)	0.75 (0.18)

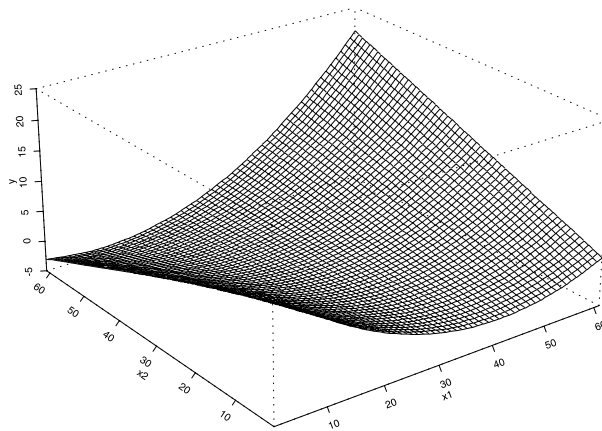


Figure 6.6. Function of the example 1. Asymmetric function $y = x_1(x_1 + x_2 + 1) + \sigma \cdot \epsilon$

Table 6.2. Results of SIR1, SIR2, and SIRpp (Example 2)

H	SIR1		SIR2		SIRpp	
	$R^2(\hat{\beta}_1)$	$R^2(\hat{\beta}_2)$	$R^2(\hat{\beta}_1)$	$R^2(\hat{\beta}_2)$	$R^2(\hat{\beta}_1)$	$R^2(\hat{\beta}_2)$
5	0.97 (0.02)	0.12 (0.14)	0.92 (0.04)	0.01 (0.10)	0.92 (0.05)	0.88 (0.11)
10	0.97 (0.02)	0.12 (0.15)	0.90 (0.06)	0.05 (0.07)	0.88 (0.08)	0.84 (0.13)
20	0.97 (0.02)	0.12 (0.14)	0.85 (0.09)	0.05 (0.06)	0.84 (0.10)	0.73 (0.22)

The SIRpp algorithm performs well in finding the e.d.r. directions; however, the algorithm requires more computing power. This is one part of projection pursuit for which the algorithm is time consuming.

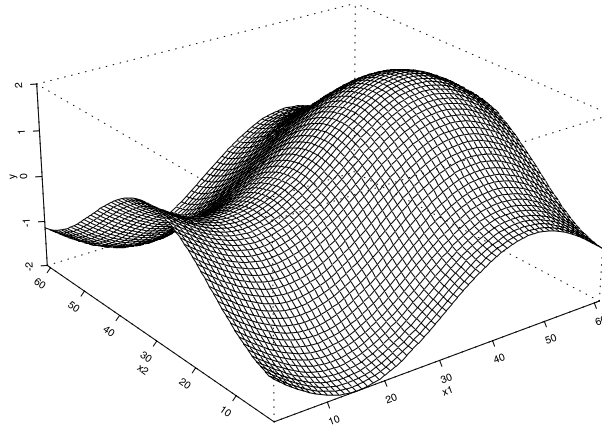


Figure 6.7. Function of the example 2. Function of asymmetric with respect to the x_1 axis and symmetric with respect to x_2 axis. $y = \sin(x_1) + \cos(x_2) + \sigma \cdot \varepsilon$

6.5

Concluding Remarks

In this chapter, we discussed dimension reduction methods for data analysis. First, PCA methods were explained for the linear method. Then, projection pursuit methods were described. For nonlinear methods, GPCA algebraic curve fitting methods and principal curves were introduced. Finally, we explained sliced inverse regression for the reduction of the dimension of explanatory variable space.

These methods are not only useful for data analysis, but also effective for preprocessing when carrying out another data analysis. In particular, they are indispensable for the analysis of enormous amounts of and complex data, e.g. microarray data, log data on the Internet, etc. Research in this field will continue to evolve in the future.

References

- Friedman, J. (1987). Exploratory projection pursuit. *Journal of the American Statistical Association*, 82:249–266.
- Friedman, J. and Tukey, J. (1974). A projection pursuit algorithm for exploratory data analysis. *IEEE Transaction on Computer*, c-23(9):881–890.
- Gnanadesikan, R. and Wilk, M. (1969). Data analytic methods. In Krishnaiah, P., editor, *Multivariate Analysis II*, pages 593–638. Academic Press.
- Hall, P. (1989). On polynomial-based projection indices for exploratory projection pursuit. *Annals of Statistics*, 17:589–605.
- Hastie, T. and Stuetzle, W. (1989). Principal curves. *Journal of the American Statistical Association*, 84:502–516.
- Huber, P. (1985). Projection pursuit (with discussion). *Annals of Statistics*, 13:435–475.

- Iwasaki, M. (1991). Projection pursuit: the idea and practice (in Japanese). *Bulletin of the Computational Statistics of Japan*, 4(2):41–56.
- Jones, M. C. and Sibson, R. (1987). What is projection pursuit? (with discussion). *Journal of the Royal Statistical Society, Series A*, 150:1–36.
- Keren, D., Cooper, D., and Subrahmonia, J. (1994). Describing complicated objects by implicit polynomials. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 16(1):38–53.
- Koyama, K., Morita, A., Mizuta, M., and Sato, Y. (1998). Projection pursuit into three dimensional space (in Japanese). *The Japanese Journal of Behaviormetrics*, 25(1):1–9.
- Li, K. (1991). Sliced inverse regression for dimension reduction. *Journal of the American Statistical Association*, 86:316–342.
- Mizuta, M. (1983). Generalized principal components analysis invariant under rotations of a coordinate system. *Journal of the Japan Statistical Society*, 14:1–9.
- Mizuta, M. (1995). A derivation of the algebraic curve for two-dimensional data using the least-squares distance. In Escoufier, Y., Hayashi, C., Fichet, B., Ohsumi, N., Diday, E., Baba, Y., and Lebart, L., editors, *Data Science and Its Application*, pages 167–176. Academic Press, Tokyo.
- Mizuta, M. (1996). Algebraic curve fitting for multidimensional data with exact squares distance. In *Proceedings of IEEE International Conference on Systems, Man and Cybernetics*, pages 516–521.
- Mizuta, M. (1997). Rasterizing algebraic curves and surfaces in the space with exact distances. In *Progress in Connectionist-Based Information Systems – Proceedings of the 1997 International Conference on Neural Information Processing and Intelligent Information Systems*, pages 551–554.
- Mizuta, M. (2002). Relative projection pursuit. In Sokolowski, A. and Jajuga, K., editors, *Data Analysis, Classification, and Related Methods*, page 131. Cracow University of Economics.
- Nason, G. (1995). Three-dimensional projection pursuit (with discussion). *Applied Statistics*, 44(4):411–430.
- Taubin, G. (1991). Estimation of planar curves, surfaces, and nonplanar space curves defined by implicit equations with applications to edge and range image segmentation. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 13(11):1115–1138.
- Taubin, G. (1994). Distance approximations for rasterizing implicit curves. *ACM Transaction on Graphics*, 13:3–42.
- Taubin, G., Cukierman, F., Sullivan, S., Ponce, J., and Kriegman, D. (1994). Parameterized families of polynomials for bounded algebraic curve and surface fitting. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 16(3):287–303.

Generalized Linear Models

III.7

Marlene Müller

7.1	<i>Introduction</i>	592
7.2	<i>Model Characteristics</i>	593
	Exponential Family	593
	Link Function	594
7.3	<i>Estimation</i>	596
	Properties of the Exponential Family	596
	Maximum-Likelihood and Deviance Minimization	597
	Iteratively Reweighted Least Squares Algorithm	598
	Remarks on the Algorithm	600
	Model Inference	602
7.4	<i>Practical Aspects</i>	603
7.5	<i>Complements and Extensions</i>	611
	Weighted Regression	611
	Overdispersion	612
	Quasi- or Pseudo-Likelihood	612
	Multinomial Responses	612
	Contingency Tables	613
	Survival Analysis	614
	Clustered Data	614
	Semiparametric Generalized Linear Models	615

7.1 Introduction

Generalized linear models (GLM) extend the concept of the well understood linear regression model. The linear model assumes that the conditional expectation of Y (the dependent or response variable) is equal to a linear combination $\mathbf{X}^\top \boldsymbol{\beta}$, i.e.

$$E(Y|\mathbf{X}) = \mathbf{X}^\top \boldsymbol{\beta}.$$

This could be equivalently written as $Y = \mathbf{X}^\top \boldsymbol{\beta} + \varepsilon$. Unfortunately, the restriction to linearity cannot take into account a variety of practical situations. For example, a continuous distribution of the error ε term implies that the response Y must have a continuous distribution as well. Hence, the linear regression model may fail when dealing with binary Y or with counts.

1 Example 1 (Bernoulli responses)

Let us illustrate a binary response model (Bernoulli Y) using a sample on credit worthiness. For each individual in the sample we know if the granted loan has defaulted or not. The responses are coded as

$$Y = \begin{cases} 1 & \text{loan defaults,} \\ 0 & \text{otherwise.} \end{cases}$$

The term of interest is how credit worthiness depends on observable individual characteristics \mathbf{X} (age, amount and duration of loan, employment, purpose of loan, etc.). Recall that for a Bernoulli variable $P(Y = 1|\mathbf{X}) = E(Y|\mathbf{X})$ holds. Hence, the default probability $P(Y = 1|\mathbf{X})$ equals a regression of Y on \mathbf{X} . A useful approach is the following *logit* model:

$$P(Y = 1|\mathbf{X} = \mathbf{x}) = \frac{1}{1 + \exp(-\mathbf{x}^\top \boldsymbol{\beta})}.$$

Here the function of interest $E(Y|\mathbf{X})$ is linked to a linear function of the explanatory variables by the logistic cumulative distribution function (cdf) $F(u) = 1/(1 + e^{-u}) = e^u/(1 + e^u)$.

The term *generalized linear models* (GLM) goes back to Nelder and Wedderburn, 1972 and McCullagh and Nelder, 1989 who show that if the distribution of the dependent variable Y is a member of the exponential family, then the class of models which connects the expectation of Y to a linear combination of the variables $\mathbf{X}^\top \boldsymbol{\beta}$ can be treated in a unified way. In the following sections we denote the function which relates $\mu = E(Y|\mathbf{X})$ and $\eta = \mathbf{X}^\top \boldsymbol{\beta}$ by $\eta = G(\mu)$ or

$$E(Y|\mathbf{X}) = G^{-1}(\mathbf{X}^\top \boldsymbol{\beta}).$$

This function G is called *link function*. For all considered distributions of Y there exists at least one canonical link function and typically a set of frequently used link functions.

Model Characteristics

7.2

The generalized linear model is determined by two components:

- the distribution of Y ,
- the link function.

In order to define the GLM methodology as a specific class of nonlinear models (for a general approach to nonlinear regression see Chap. III.8), we assume that the distribution of Y is a member of the *exponential family*. The exponential family covers a large number of distributions, for example discrete distributions as the Bernoulli, binomial and Poisson which can handle binary and count data or continuous distributions as the normal, Gamma or Inverse Gaussian distribution.

Exponential Family

7.2.1

We say that a distribution is a member of the exponential family if its probability mass function (if Y discrete) or its density function (if Y continuous) has the following form:

$$f(y, \theta, \psi) = \exp \left\{ \frac{y\theta - b(\theta)}{a(\psi)} + c(y, \psi) \right\}. \quad (7.1)$$

The functions $a(\bullet)$, $b(\bullet)$ and $c(\bullet)$ will vary for different Y distributions. Our parameter of interest is θ , which is also called the *canonical parameter* (McCullagh and Nelder, 1989). The additional parameter ψ , that is only relevant for some of the distributions, is considered as a nuisance parameter.

Example 2 (Normal distribution)

2

Suppose Y is normally distributed with $Y \sim N(\mu, \sigma^2)$. The probability density function $f(y) = \exp \{ -(y - \mu)^2 / (2\sigma^2) \} / (\sqrt{2\pi}\sigma)$ can be written as in (7.1) by setting $\theta = \mu$ and $\psi = \sigma$ and $a(\psi) = \psi^2$, $b(\theta) = \theta^2/2$, and $c(y, \psi) = -y^2/(2\psi^2) - \log(\sqrt{2\pi}\psi)$.

Example 3 (Bernoulli distribution)

3

If Y is Bernoulli distributed its probability mass function is

$$P(Y = y) = \mu^y (1 - \mu)^{1-y} = \begin{cases} \mu & \text{if } y = 1, \\ 1 - \mu & \text{if } y = 0. \end{cases}$$

This can be transformed into $P(Y = y) = \exp(y\theta) / (1 + e^\theta)$ using the *logit* transformation $\theta = \log \{ \mu / (1 - \mu) \}$ equivalent to $\mu = e^\theta / (1 + e^\theta)$. Thus we obtain an exponential family with $a(\psi) = 1$, $b(\theta) = -\log(1 - \mu) = \log(1 + e^\theta)$, and $c(y, \psi) = 0$.

Table 7.1. GLM distributions

	Range of y	$f(y)$	$\mu(\theta)$	Variance terms	
				$V(\mu)$	$a(\psi)$
Bernoulli $B(\mu)$	$\{0, 1\}$	$\mu^y(1-\mu)^{1-y}$	$\frac{e^\theta}{1+e^\theta}$	$\mu(1-\mu)$	1
Binomial $B(k, \mu)$	$\{0, \dots, k\}$	$\binom{k}{y} \mu^y(1-\mu)^{k-y}$	$\frac{ke^\theta}{1+e^\theta}$	$\mu\left(1-\frac{\mu}{k}\right)$	1
Poisson $P(\mu)$	$\{0, 1, 2, \dots\}$	$\frac{\mu^y}{y!} e^{-\mu}$	$\exp(\theta)$	μ	1
Geometric $Geo(\mu)$	$\{0, 1, 2, \dots\}$	$\left(\frac{\mu}{1+\mu}\right)^y \left(\frac{1}{1+\mu}\right)$	$\frac{e^\theta}{1-e^\theta}$	$\mu + \mu^2$	1
Negative Binomial $NB(\mu, k)$	$\{0, 1, 2, \dots\}$	$\binom{k+y-1}{y} \left(\frac{\mu}{k+\mu}\right)^y \left(\frac{k}{k+\mu}\right)$	$\frac{ke^\theta}{1-e^\theta}$	$\mu + \frac{\mu^2}{k}$	1
Exponential $Exp(\mu)$	$(0, \infty)$	$\frac{1}{\mu} \exp\left(-\frac{x}{\mu}\right)$	$-1/\theta$	μ^2	1
Gamma $G(\mu, \psi)$	$(0, \infty)$	$\frac{1}{\Gamma(\psi)} \left(\frac{\psi}{\mu}\right)^\psi \exp\left(-\frac{\psi y}{\mu}\right) y^{\psi-1}$	$-1/\theta$	μ^2	$\frac{1}{\psi}$
Normal $N(\mu, \psi^2)$	$(-\infty, \infty)$	$\frac{\exp\{-\frac{(y-\mu)^2}{2\psi^2}\}}{\sqrt{2\pi\psi}}$	θ	1	ψ^2
Inverse Gaussian $IG(\mu, \psi^2)$	$(0, \infty)$	$\frac{\exp\{-\frac{(y-\mu)^2}{2\mu^2\psi^2}\}}{\sqrt{2\pi y^3\psi}}$	$\frac{1}{\sqrt{-2\theta}}$	μ^3	ψ^2

Table 7.1 lists some probability distributions that are typically used for a GLM. For the binomial and negative binomial distribution the additional parameter k is assumed to be known. Note also that the Bernoulli, geometric and exponential distributions are special cases of the binomial, negative binomial and Gamma distributions, respectively.

7.2.2 Link Function

After having specified the distribution of Y , the link function G is the second component to choose for the GLM. Recall the model notation $\eta = \mathbf{X}^\top \boldsymbol{\beta} = G(\mu)$. In the case that the canonical parameter θ equals the linear predictor η , i.e. if

$$\eta = \theta,$$

the link function is called the *canonical link* function. For models with a canonical link the estimation algorithm simplifies as we will see in Sect. 7.3.3. Table 7.2 shows in its second column the canonical link functions of the exponential family distributions presented in Table 7.1.

Table 7.2. Characteristics of GLMs

	Canonical link $\theta(\mu)$	Deviance $D(y, \mu)$
Bernoulli $B(\mu)$	$\log\left(\frac{\mu}{1-\mu}\right)$	$2 \sum \left[y_i \log\left(\frac{y_i}{\mu_i}\right) + (1 - y_i) \log\left(\frac{1 - y_i}{1 - \mu_i}\right) \right]$
Binomial $B(k, \mu)$	$\log\left(\frac{\mu}{k - \mu}\right)$	$2 \sum \left[y_i \log\left(\frac{y_i}{\mu_i}\right) + (k - y_i) \log\left(\frac{k - y_i}{k - \mu_i}\right) \right]$
Poisson $P(\mu)$	$\log(\mu)$	$2 \sum \left[y_i \log\left(\frac{y_i}{\mu_i}\right) - (y_i - \mu_i) \right]$
Geometric $Geo(\mu)$	$\log\left(\frac{\mu}{1 + \mu}\right)$	$2 \sum \left[y_i \log\left(\frac{y_i + y_i \mu_i}{\mu_i + y_i \mu_i}\right) - \log\left(\frac{1 + y_i}{1 + \mu_i}\right) \right]$
Negative Binomial $NB(\mu, k)$	$\log\left(\frac{\mu}{k + \mu}\right)$	$2 \sum \left[y_i \log\left(\frac{y_i k + y_i \mu_i}{\mu_i k + y_i \mu_i}\right) - k \log\left\{ \frac{k(k + y_i)}{k(k + \mu_i)} \right\} \right]$
Exponential $Exp(\mu)$	$\frac{1}{\mu}$	$2 \sum \left[\frac{y_i - \mu_i}{\mu_i} - \log\left(\frac{y_i}{\mu_i}\right) \right]$
Gamma $G(\mu, \psi)$	$\frac{1}{\mu}$	$2 \sum \left[\frac{y_i - \mu_i}{\mu_i} - \log\left(\frac{y_i}{\mu_i}\right) \right]$
Normal $N(\mu, \psi^2)$	μ	$2 \sum \left[(y_i - \mu_i)^2 \right]$
Inverse Gaussian $IG(\mu, \psi^2)$	$\frac{1}{\mu^2}$	$2 \sum \left[\frac{(y_i - \mu_i)^2}{y_i \mu_i^2} \right]$

Example 4 (Canonical link for Bernoulli Y)

For Bernoulli Y we have $\mu = e^\theta / (1 + e^\theta)$, hence the canonical link is given by the logit transformation $\eta = \log\{\mu / (1 - \mu)\}$.

What link functions could we choose apart from the canonical? For most of the models exists a number of specific link functions. For Bernoulli Y , for example, any smooth cdf can be used. Typical links are the logistic and standard normal

(Gaussian) cdfs which lead to logit and *probit* models, respectively. A further alternative for Bernoulli Y is the complementary log–log link $\eta = \log\{-\log(1-\mu)\}$.

A flexible class of link functions for positive Y observations is the class of power functions. These links are given by the Box–Cox transformation (Box and Cox, 1964), i.e. by $\eta = (\mu^\lambda - 1)/\lambda$ or $\eta = \mu^\lambda$ where we set in both cases $\eta = \log(\mu)$ for $\lambda = 0$.

7.3 Estimation

Recall that the least squares estimator for the ordinary linear regression model is also the maximum-likelihood estimator in the case of normally distributed error terms. By assuming that the distribution of Y belongs to the exponential family it is possible to derive maximum-likelihood estimates for the coefficients of a GLM. Moreover we will see that even though the estimation needs a numerical approximation, each step of the iteration can be given by a weighted least squares fit. Since the weights are varying during the iteration the likelihood is optimized by an *iteratively reweighted least squares* algorithm.

7.3.1 Properties of the Exponential Family

To derive the details of the maximum-likelihood algorithm we need to discuss some properties of the probability mass or density function $f(\bullet)$. For the sake of brevity we consider f to be a density function in the following derivation. However, the conclusions will hold for a probability mass function as well.

First, we start from the fact that $\int f(y, \theta, \psi) dy = 1$. Under suitable regularity conditions (it is possible to exchange differentiation and integration) this implies

$$\begin{aligned} 0 &= \frac{\partial}{\partial \theta} \int f(y, \theta, \psi) dy = \int \frac{\partial}{\partial \theta} f(y, \theta, \psi) dy \\ &= \int \left\{ \frac{\partial}{\partial \theta} \log f(y, \theta, \psi) \right\} f(y, \theta, \psi) dy = E \left\{ \frac{\partial}{\partial \theta} \ell(y, \theta, \psi) \right\}, \end{aligned}$$

where $\ell(y, \theta, \psi) = \log f(y, \theta, \psi)$ denotes the *log-likelihood* function. The function derivative of ℓ with respect to θ is typically called the *score* function for which it is known that

$$E \left\{ \frac{\partial^2}{\partial \theta^2} \ell(y, \theta, \psi) \right\} = -E \left\{ \frac{\partial}{\partial \theta} \ell(y, \theta, \psi) \right\}^2.$$

This and taking first and second derivatives of (7.1) results in

$$0 = E \left\{ \frac{Y - b'(\theta)}{a(\psi)} \right\}, \quad \text{and} \quad E \left\{ \frac{-b''(\theta)}{a(\psi)} \right\} = -E \left\{ \frac{Y - b'(\theta)}{a(\psi)} \right\}^2,$$

such that we can conclude

$$E(Y) = \mu = b'(\theta), \tag{7.2}$$

$$\text{Var}(Y) = V(\mu)a(\psi) = b''(\theta)a(\psi). \tag{7.3}$$

Note that as a consequence from (7.1) the expectation of Y depends only on the parameter of interest θ . We also assume that the factor $a(\psi)$ is identical over all observations.

Maximum-Likelihood and Deviance Minimization

7.3.2

As pointed out before the estimation method of choice for β is maximum-likelihood. As an alternative the literature refers to the minimization of the *deviance*. We will see during the following derivation that both approaches are identical.

Suppose that we have observed a sample of independent pairs (Y_i, X_i) where $i = 1, \dots, n$. For a more compact notation denote now the vector of all response observations by $\mathbf{Y} = (Y_1, \dots, Y_n)^\top$ and their conditional expectations (given X_i) by $\boldsymbol{\mu} = (\mu_1, \dots, \mu_n)^\top$. Recall that we study

$$E(Y_i|X_i) = \mu_i = G(\mathbf{X}_i^\top \boldsymbol{\beta}) = G(\eta_i).$$

The sample log-likelihood of the vector \mathbf{Y} is then given by

$$\ell(\mathbf{Y}, \boldsymbol{\mu}, \psi) = \sum_{i=1}^n \ell(Y_i, \theta_i, \psi). \quad (7.4)$$

Here θ_i is a function of $\eta_i = \mathbf{X}_i^\top \boldsymbol{\beta}$ and we use $\ell(Y_i, \theta_i, \psi) = \log f(Y_i, \theta_i, \psi)$ to denote the individual log-likelihood contributions for all observations i .

Example 5 (Normal log-likelihood)

5

For normal responses $Y_i \sim N(\mu_i, \sigma^2)$ we have $\ell(Y_i, \theta_i, \psi) = -(Y_i - \mu_i)^2 / (2\sigma^2) - \log(\sqrt{2\pi}\sigma)$. This gives the sample log-likelihood

$$\ell(\mathbf{Y}, \boldsymbol{\mu}, \sigma) = n \log \left(\frac{1}{\sqrt{2\pi}\sigma} \right) - \frac{1}{2\sigma^2} \sum_{i=1}^n (Y_i - \mu_i)^2. \quad (7.5)$$

Obviously, maximizing this log-likelihood is equivalent to minimizing the least squares criterion.

Example 6 (Bernoulli log-likelihood)

6

The calculation in Example 3 shows that the individual log-likelihoods for the binary responses equal $\ell(Y_i, \theta_i, \psi) = Y_i \log(\mu_i) + (1 - Y_i) \log(1 - \mu_i)$. This leads to

$$\ell(\mathbf{Y}, \boldsymbol{\mu}, \psi) = \sum_{i=1}^n \{Y_i \log(\mu_i) + (1 - Y_i) \log(1 - \mu_i)\} \quad (7.6)$$

for the sample version.

The deviance defines an alternative objective function for optimization. Let us first introduce the *scaled deviance* which is defined as

$$D(\mathbf{Y}, \boldsymbol{\mu}, \psi) = 2 \left\{ \ell(\mathbf{Y}, \boldsymbol{\mu}^{\max}, \psi) - \ell(\mathbf{Y}, \boldsymbol{\mu}, \psi) \right\}. \quad (7.7)$$

Here $\boldsymbol{\mu}^{\max}$ (which typically equals \mathbf{Y}) is the vector that maximizes the saturated model, i.e. the function $\ell(\mathbf{Y}, \boldsymbol{\mu}, \psi)$ without imposing any restriction on $\boldsymbol{\mu}$. Since the term $\ell(\mathbf{Y}, \boldsymbol{\mu}^{\max}, \psi)$ does not depend on the parameter $\boldsymbol{\beta}$ we see that indeed the minimization of the scaled deviance is equivalent to the maximization of the sample log-likelihood (7.4).

If we now plug-in the exponential family form (7.1) into (7.4) we obtain

$$\ell(\mathbf{Y}, \boldsymbol{\mu}, \psi) = \sum_{i=1}^n \left\{ \frac{Y_i \theta_i - b(\theta_i)}{a(\psi)} - c(Y_i, \psi) \right\}. \quad (7.8)$$

Obviously, neither $a(\psi)$ nor $c(Y_i, \psi)$ depend on the unknown parameter vector $\boldsymbol{\beta}$. Therefore, it is sufficient to consider

$$\sum_{i=1}^n \{Y_i \theta_i - b(\theta_i)\} \quad (7.9)$$

for the maximization. The deviance analog of (7.9) is the (non-scaled) deviance function

$$D(\mathbf{Y}, \boldsymbol{\mu}) = D(\mathbf{Y}, \boldsymbol{\mu}, \psi) a(\psi). \quad (7.10)$$

The (non-scaled) deviance $D(\mathbf{Y}, \boldsymbol{\mu})$ can be seen as the GLM equivalent of the *residual sum of squares* (RSS) in linear regression as it compares the log-likelihood ℓ for the “model” $\boldsymbol{\mu}$ with the maximal achievable value of ℓ .

7.3.3 Iteratively Reweighted Least Squares Algorithm

We will now minimize the deviance with respect to $\boldsymbol{\beta}$. If we denote the gradient of (7.10) by

$$\nabla(\boldsymbol{\beta}) = \frac{\partial}{\partial \boldsymbol{\beta}} \left[-2 \sum_{i=1}^n \{Y_i \theta_i - b(\theta_i)\} \right] = -2 \sum_{i=1}^n \{Y_i - b'(\theta_i)\} \frac{\partial}{\partial \boldsymbol{\beta}} \theta_i, \quad (7.11)$$

our optimization problem consists in solving

$$\nabla(\boldsymbol{\beta}) = 0. \quad (7.12)$$

Note that this is (in general) a nonlinear system of equations in $\boldsymbol{\beta}$ and an iterative solution has to be computed. The smoothness of the link function allows us to compute the *Hessian* of $D(\mathbf{Y}, \boldsymbol{\mu})$, which we denote by $\mathcal{H}(\boldsymbol{\beta})$. Now a *Newton-Raphson* algorithm can be applied which determines the optimal $\hat{\boldsymbol{\beta}}$ using the following iteration steps:

$$\hat{\boldsymbol{\beta}}^{\text{new}} = \hat{\boldsymbol{\beta}}^{\text{old}} - \left\{ \mathcal{H}(\hat{\boldsymbol{\beta}}^{\text{old}}) \right\}^{-1} \nabla \left(\hat{\boldsymbol{\beta}}^{\text{old}} \right).$$

A variant of the Newton–Raphson is the *Fisher scoring* algorithm that replaces the Hessian by its expectation with respect to the observations Y_i :

$$\hat{\boldsymbol{\beta}}^{\text{new}} = \hat{\boldsymbol{\beta}}^{\text{old}} - \left\{ E\mathcal{H}(\hat{\boldsymbol{\beta}}^{\text{old}}) \right\}^{-1} \nabla \left(\hat{\boldsymbol{\beta}}^{\text{old}} \right).$$

To find simpler representations for these iterations, recall that we have $\mu_i = G(\eta_i) = G(\mathbf{X}_i^\top \boldsymbol{\beta}) = b'(\theta_i)$. By taking the derivative of the right hand term with respect to $\boldsymbol{\beta}$ this implies

$$b'(\theta_i) \frac{\partial}{\partial \boldsymbol{\beta}} \theta_i = G(\mathbf{X}_i^\top \boldsymbol{\beta}) \mathbf{X}_i.$$

Using that $b''(\theta_i) = V(\mu_i)$ as established in (7.3) and taking derivatives again, we finally obtain

$$\frac{\partial}{\partial \boldsymbol{\beta}} \theta_i = \frac{G'(\eta_i)}{V(\mu_i)} \mathbf{X}_i$$

$$\frac{\partial^2}{\partial \boldsymbol{\beta} \boldsymbol{\beta}^\top} \theta_i = \frac{G''(\eta_i)V(\mu_i) - G'(\eta_i)^2 V'(\mu_i)}{V(\mu_i)^2} \mathbf{X}_i \mathbf{X}_i^\top.$$

From this we can express the gradient and the Hessian of the deviance by

$$\nabla(\boldsymbol{\beta}) = -2 \sum_{i=1}^n \{Y_i - \mu_i\} \frac{G'(\eta_i)}{V(\mu_i)} \mathbf{X}_i$$

$$\mathcal{H}(\boldsymbol{\beta}) = 2 \sum_{i=1}^n \left\{ \frac{G'(\eta_i)^2}{V(\mu_i)} - \{Y_i - \mu_i\} \frac{G''(\eta_i)V(\mu_i) - G'(\eta_i)^2 V'(\mu_i)}{V(\mu_i)^2} \right\} \mathbf{X}_i \mathbf{X}_i^\top.$$

The expectation of $\mathcal{H}(\boldsymbol{\beta})$ in the Fisher scoring algorithm equals

$$E\mathcal{H}(\boldsymbol{\beta}) = 2 \sum_{i=1}^n \left\{ \frac{G'(\eta_i)^2}{V(\mu_i)} \right\} \mathbf{X}_i \mathbf{X}_i^\top.$$

Let us consider only the Fisher scoring algorithm for the moment. We define the weight matrix

$$\mathbf{W} = \text{diag} \left(\frac{G'(\eta_1)^2}{V(\mu_1)}, \dots, \frac{G'(\eta_n)^2}{V(\mu_n)} \right)$$

and the vectors $\tilde{\mathbf{Y}} = (\tilde{Y}_1, \dots, \tilde{Y}_n)^\top$, $\mathbf{Z} = (Z_1, \dots, Z_n)^\top$ by

$$\tilde{Y}_i = \frac{Y_i - \mu_i}{G'(\eta_i)}, \quad Z_i = \eta_i + \tilde{Y}_i = \mathbf{X}_i^\top \boldsymbol{\beta}^{\text{old}} + \frac{Y_i - \mu_i}{G'(\eta_i)}.$$

Denote further by X the design matrix given by the rows x_i^\top . Then, the Fisher scoring iteration step for β can be rewritten as

$$\beta^{\text{new}} = \beta^{\text{old}} + (X^\top W X)^{-1} X^\top W \tilde{Y} = (X^\top W X)^{-1} X^\top W Z. \quad (7.13)$$

This immediately shows that each Fisher scoring iteration step is the result of a weighted least squares regression of the *adjusted dependent variables* Z_i on the explanatory variables X_i . Since the weights are recalculated in each step we speak of the *iteratively reweighted least squares* (IRLS) algorithm. For the Newton–Raphson algorithm a representation equivalent to (7.13) can be found, only the weight matrix W differs.

The iteration will be stopped when the parameter estimate and/or the deviance do not change significantly anymore. We denote the final parameter estimate by $\tilde{\beta}$.

7.3.4 Remarks on the Algorithm

Let us first note two special cases for the algorithm:

- In the linear regression model, where we have $G' \equiv 1$ and $\mu_i = \eta_i = X_i^\top \beta$, no iteration is necessary. Here the ordinary least squares estimator gives the explicit solution of (7.12).
- In the case of a canonical link function we have $b'(\theta_i) = G(\theta_i) = G(\eta_i)$ and hence $b''(\theta_i) = G'(\eta_i) = V(\mu_i)$. Therefore the Newton–Raphson and the Fisher scoring algorithms coincide.

There are several further remarks on the algorithm which concern in particular starting values and the computation of relevant terms for the statistical analysis:

- Equation (7.13) implies that in fact we do not need a starting value for β . Indeed the adjusted dependent variables Z_i can be equivalently initialized by using appropriate values for $\eta_{i,0}$ and $\mu_{i,0}$. Typically, the following initialization is used (McCullagh and Nelder, 1989):
 - * For all but binomial models set $\mu_{i,0} = Y_i$ and $\eta_{i,0} = G(\mu_{i,0})$.
 - * For binomial models set $\mu_{i,0} = (Y_i + \frac{1}{2})/(k + 1)$ and $\eta_{i,0} = G(\mu_{i,0})$. (Recall that this holds with $k = 1$ in the Bernoulli case.)

The latter definition is based on the observation that G can not be applied to binary data. Therefore a kind of smoothing is used to obtain $\mu_{i,0}$ in the binomial case.

- During the iteration the convergence can be controlled by checking the relative change in the coefficients

$$\sqrt{\frac{(\beta^{\text{new}} - \beta^{\text{old}})^\top (\beta^{\text{new}} - \beta^{\text{old}})}{\beta^{\text{old}\top} \beta^{\text{old}}}} < \varepsilon$$

and/or the relative change in the deviance

$$\left| \frac{D(\mathbf{Y}, \boldsymbol{\mu}^{\text{new}}) - D(\mathbf{Y}, \boldsymbol{\mu}^{\text{old}})}{D(\mathbf{Y}, \boldsymbol{\mu}^{\text{old}})} \right| < \varepsilon.$$

- An estimate $\widehat{\psi}$ for the dispersion parameter ψ can be obtained from either the Pearson χ^2 statistic

$$\widehat{a}(\psi) = \frac{1}{n-p} \sum_{i=1}^n \frac{(Y_i - \widehat{\mu}_i)^2}{V(\widehat{\mu}_i)}, \tag{7.14}$$

or using deviance

$$\widehat{a}(\psi) = \frac{D(\mathbf{Y}, \boldsymbol{\mu})}{n-p}. \tag{7.15}$$

Here we use p for the number of estimated parameters and $\widehat{\mu}_i$ for the estimated regression function at the i th observation. Similarly, $\widehat{\boldsymbol{\mu}}$ is the estimated $\boldsymbol{\mu}$. Both estimators for $a(\psi)$ coincide for normal linear regression and follow an exact χ^2_{n-p} distribution then. The number $n - p$ (number of observations minus number of estimated parameters) is denoted as the *degrees of freedom* of the deviance.

- Typically, software for GLM allows for offsets and weights in the model. For details on the inclusion of weights we refer to Sect. 7.5.1. Offsets are deterministic components of $\boldsymbol{\eta}$ which can vary over the observations i . The model that is then fitted is

$$E(Y_i | \mathbf{X}_i) = G(\mathbf{X}_i^\top \boldsymbol{\beta} + o_i).$$

Offsets may be used to fit a model where a part of the coefficients is known. The iteration algorithm stays unchanged except for the fact that the optimization is only necessary with respect to the remaining unknown coefficients.

- Since the variance of Y_i will usually depend on \mathbf{X}_i we cannot simply analyze residuals of the form $Y_i - \widehat{\mu}_i$. Instead, appropriate transformations have to be used. Classical proposals are Pearson residuals

$$r_i^P = \frac{Y_i - \widehat{\mu}_i}{\sqrt{V(\widehat{\mu}_i)}},$$

deviance residuals

$$r_i^D = \text{sign}(Y_i - \widehat{\mu}_i) \sqrt{d_i},$$

where d_i is the contribution of the i th observation to the deviance, and Anscombe residuals

$$r_i^A = \frac{A(Y_i) - A(\widehat{\mu}_i)}{A'(\widehat{\mu}_i) \sqrt{V(\widehat{\mu}_i)}},$$

where $A(\boldsymbol{\mu}) = \int^\boldsymbol{\mu} V^{-1/3}(u) du$.

7.3.5 Model Inference

The resulting estimator $\widehat{\boldsymbol{\beta}}$ has an asymptotic normal distribution (except of course for the normal linear regression case when this is an exact normal distribution).

1

Theorem 1

Under regularity conditions we have for the estimated coefficient vector

$$\sqrt{n}(\widehat{\boldsymbol{\beta}} - \boldsymbol{\beta}) \rightarrow N(0, \boldsymbol{\Sigma}) \quad \text{as } n \rightarrow \infty.$$

As a consequence for the scaled deviance and the log-likelihood approximately hold $D(\mathbf{Y}, \widehat{\boldsymbol{\mu}}, \psi) \sim \chi_{n-p}^2$ and $2\{\ell(\mathbf{Y}, \widehat{\boldsymbol{\mu}}, \psi) - \ell(\mathbf{Y}, \boldsymbol{\mu}, \psi)\} \sim \chi_p^2$.

For details on the necessary conditions see for example Fahrmeir and Kaufmann, 1984. Note also that the asymptotic covariance $\boldsymbol{\Sigma}$ for the coefficient estimator $\widehat{\boldsymbol{\beta}}$ is the inverse of the Fisher information matrix, i.e.

$$\mathbf{I} = -E \left\{ \frac{\partial^2}{\partial \boldsymbol{\beta} \boldsymbol{\beta}^T} \ell(\mathbf{Y}, \boldsymbol{\mu}, \psi) \right\}.$$

Since \mathbf{I} can be estimated by the negative Hessian of the log-likelihood or its expectation, this suggests the estimator

$$\widehat{\boldsymbol{\Sigma}} = a(\widehat{\psi}) \left[\frac{1}{n} \sum_{i=1}^n \left\{ \frac{G'(\eta_{i,\text{last}})^2}{V(\boldsymbol{\mu}_{i,\text{last}})} \right\} \mathbf{X}_i \mathbf{X}_i^T \right]^{-1}.$$

Using the estimated covariance we are able to test hypotheses about the components of $\boldsymbol{\beta}$.

For model choice between two nested models a likelihood ratio test (LR test) is used. Assume that \mathcal{M}_0 (p_0 parameters) is a submodel of the model \mathcal{M} (p parameters) and that we have estimated them as $\widehat{\boldsymbol{\mu}}_0$ and $\widehat{\boldsymbol{\mu}}$. For one-parameter exponential families (without a nuisance parameter ψ) we use that asymptotically

$$D(\mathbf{Y}, \boldsymbol{\mu}_0) - D(\mathbf{Y}, \boldsymbol{\mu}) \sim \chi_{p-p_0}^2. \quad (7.16)$$

The left hand side of (7.16) is a function of the ratio of the two likelihoods deviance difference equals minus twice the log-likelihood difference. In a two-parameter exponential family (ψ is to be estimated) one can approximate the likelihood ratio test statistic by

$$\frac{(n-p)\{D(\mathbf{Y}, \boldsymbol{\mu}_0) - D(\mathbf{Y}, \boldsymbol{\mu})\}}{(p-p_0)D(\mathbf{Y}, \boldsymbol{\mu})} \sim F_{p-p_0, n-p} \quad (7.17)$$

using the analog to the normal linear regression case (Venables and Ripley, 2002), Chap. 7.

Model selection procedures for possibly non-nested models can be based on Akaike's information criterion (Akaike, 1973)

Table 7.3. Credit data

Variable	Yes	No	(in %)	
Y (observed default)	30.0	70.0		
PREVIOUS (no problem)	38.1	61.9		
EMPLOYED (≥ 1 year)	93.8	6.2		
DURATION (9, 12]	21.6	78.4		
DURATION (12, 18]	18.7	81.3		
DURATION (18, 24]	22.4	77.6		
DURATION ≥ 24	23.0	77.0		
SAVINGS	18.3	81.7		
PURPOSE (buy a car)	28.4	71.6		
HOUSE (owner)	15.4	84.6		
	Min.	Max.	Mean	Std.Dev.
AMOUNT (in DM)	250	18424	3271.248	2822.752
AGE (in years)	19	75	35.542	11.353

$$AIC = D(Y, \hat{\boldsymbol{\mu}}, \hat{\boldsymbol{\psi}}) + 2p,$$

or Schwarz' Bayes information criterion (Schwarz, 1978)

$$BIC = D(Y, \hat{\boldsymbol{\mu}}, \hat{\boldsymbol{\psi}}) + \log(n)p,$$

where again p denotes the number of estimated parameters. For a general overview on model selection techniques see also Chap. III.1 of this handbook.

Practical Aspects

7.4

To illustrate the GLM in practice we recall Example 1 on credit worthiness. The credit data set that we use (Fahrmeir and Tutz, 1994) contains $n = 1000$ observations on consumer credits and a variety of explanatory variables. We have selected a subset of eight explanatory variables for the following examples.

The model for credit worthiness is based on the idea that default can be predicted from the individual and loan characteristics. We consider criteria as age, information on previous loans, savings, employment and house ownership to characterize the credit applicants. Amount and duration of the loan are prominent features of the granted loans. Some descriptive statistics can be found in Table 7.3. We remark that we have categorized the durations (months) into intervals since most of the realizations are multiples of 3 or 6 months.

We are at the first place interested in estimating the probability of credit default in dependence of the explanatory variables X . Recall that for binary Y it holds

$P(Y = 1|X) = E(Y|X)$. Our first approach is a GLM with logit link such that $P(Y = 1|X) = \exp(X^T \boldsymbol{\beta}) / \{1 + \exp(X^T \boldsymbol{\beta})\}$.

7

Example 7 (Credit default on AGE)

We initially estimate the default probability solely related to age, i.e. the model

$$P(Y = 1|AGE) = \frac{\exp(\beta_0 + \beta_1 AGE)}{1 + \exp(\beta_0 + \beta_1 AGE)}$$

or equivalently logit $\{P(Y = 1|AGE)\} = \beta_0 + \beta_1 AGE$. The resulting estimates of the constant β_0 and the slope parameter β_1 are displayed in Table 7.4 together with summary statistics on the model fit.

From the table we see that the estimated coefficient of AGE has a negative sign. Since the link function and its inverse are strictly monotone increasing, we can conclude that the probability of default must thus be decreasing with increasing AGE. Figure 7.1 shows on the left frequency barplots of AGE separately for $Y = 1$ and $Y = 0$. From the observed frequencies we can recognize clearly the decreasing propensity to default. The right graph in Fig. 7.1 displays the estimated probabilities $P(Y = 1|AGE)$ using the fitted logit model which are indeed decreasing.

The t -values $(\sqrt{n} \widehat{\beta}_j / \sqrt{\widehat{\Sigma}_{jj}})$ show that the coefficient of AGE is significantly different from 0 while the estimated constant is not. The test that is used here is an approximative t -test such that $z_{1-\alpha/2}$ -quantile of the standard normal can be used as critical value. This implies that at the usual 5% level we compare the absolute value of the t -value with $z_{0.975} \approx 1.96$.

A more general approach to test for the significance of AGE is to compare the fitted model with a model that involves only a constant default probability. Typically software packages report the deviance of this model as null deviance or similar. In our case we find a null deviance of 1221.7 at 999 degrees of freedom. If we apply the LR test statistic (7.16) to compare the null deviance to the model deviance of 1213.1 at 998 degrees of freedom, we find that constant model is clearly rejected at a significance level of 0.33%.

Table 7.4. Credit default on AGE (logit model)

Variable	Coefficient	t -value
constant	-0.1985	-0.851
AGE	-0.0185	-2.873
Deviance	1213.1	
df	998	
AIC	1217.1	
Iterations	4	

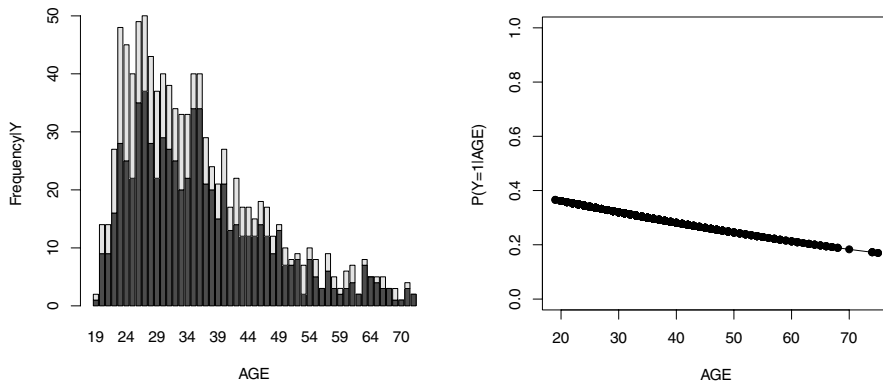


Figure 7.1. Credit default on AGE, left: frequency barplots of AGE for $Y = 1$ (light) and $Y = 0$ (dark), right: estimated probabilities

Models using different link functions cannot be directly compared as the link functions might be differently scaled. In our binary response model for example a logit or a probit link function may be reasonable. However, the variance parameter of the standard logistic distribution is $\pi^2/3$ whereas that of the standard normal is 1. We therefore need to rescale one of the link functions in order to compare the resulting model fits. Figure 7.2 shows the standard logistic cdf (the inverse logit link) against the cdf of $N(0, \pi^2/3)$. The functions in the left graph of Fig. 7.2 are hardly distinguishable. If we zoom in (right graph) we see that the logistic cdf vanishes to zero at the left boundary at a lower rate. This holds similarly for the right boundary and explains the ability of logit models to (slightly) better handle the case of extremal observations.

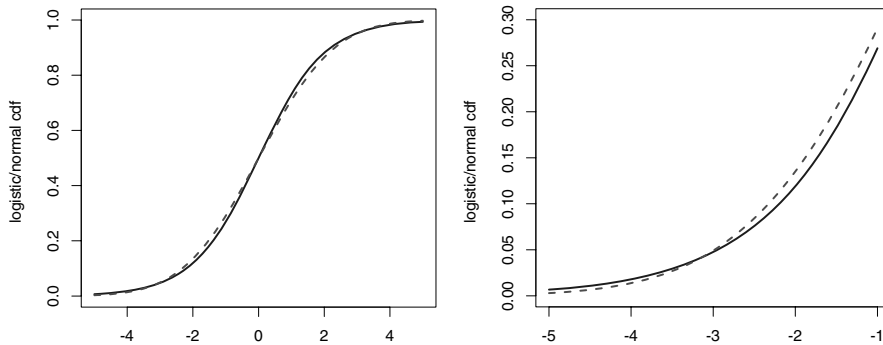


Figure 7.2. Logit (solid) versus appropriately rescaled probit link (dashed), left: on the range $[-5, 5]$, right: on the range of $[-5, -1]$

8 Example 8 (Probit versus logit)

If we want to compare the estimated coefficients from a probit to that of the logit model we need to rescale the probit coefficients by $\pi/\sqrt{3}$. Table 7.5 shows the results of a probit for credit default on AGE. The resulting rescaled coefficient for AGE is of similar size as that for the logit model (see Table 7.4) while the constant is not significantly different from 0 in both fits. The deviance and the AIC of the probit fit are slightly larger.

A Newton–Raphson iteration (instead of the Fisher scoring reported in Table 7.5) does give somewhat different coefficients but returns nearly the same value of the deviance (1213.268 for Newton–Raphson versus 1213.265 for Fisher scoring).

Table 7.5. Credit default on AGE (probit model), original and rescaled coefficients for comparison with logit

Variable	Coefficient		<i>t</i> -value
	(original)	(rescaled)	
constant	-0.1424	-0.2583	-1.022
AGE	-0.0109	-0.0197	-2.855
Deviance	1213.3		
df	998		
AIC	1217.3		
Iterations	4	(Fisher Scoring)	

The next two examples intend to analyze if the fit could be improved by using a nonlinear function on AGE instead of $\eta = \beta_0 + \beta_1 \text{AGE}$. Two principally different approaches are possible:

- include higher order terms of AGE into η ,
- categorize AGE in order to fit a stepwise constant η function.

9 Example 9 (Credit default on polynomial AGE)

We fit two logit models using second and third order terms in AGE. The estimated coefficients are presented in Table 7.6. A comparison of the quadratic fit and the linear fit from Example 7 using the LR test statistic (7.16) shows that the linear fit is rejected at a significance level of 3%. A subsequent comparison of the quadratic against the cubic fit no significant improvement by the latter model. Thus, the quadratic term for AGE improves the fit whereas the cubic term does not show any further statistically significant improvement. This result is confirmed when we compare the AIC values of both models which are practically identical. Figure 7.3 shows the estimated default probabilities for the quadratic (left) and cubic AGE fits. We find that the curves are of similar shape.

Table 7.6. Credit default on polynomial AGE (logit model)

Variable	Coefficient	<i>t</i> -value	Coefficient	<i>t</i> -value
constant	1.2430	1.799	0.4092	1.909
AGE	-0.0966	-2.699	-0.3240	-1.949
AGE**2	9.56×10^{-4}	2.234	6.58×10^{-3}	1.624
AGE**3	-	-	-4.33×10^{-5}	-1.390
Deviance	1208.3		1206.3	
df	997		996	
AIC	1214.3		1214.3	
Iterations	4		4	

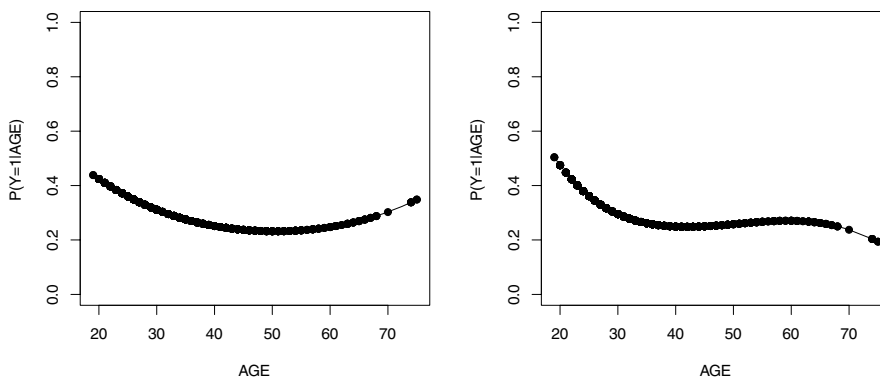


Figure 7.3. Credit default on polynomial AGE, *left*: estimated probabilities from quadratic function, *right*: estimated probabilities from cubic function

To incorporate a possible nonlinear impact of a variable in the index function, we can alternatively categorize this variable. Another term for this is the construction of *dummy* variables. The most classical form of the categorization consists in using a design matrix that sets a value of 1 in the column corresponding to the category if the category is true and 0 otherwise. To obtain a full rank design matrix we omit one column for the reference category. In our example we leave out the first category which means that all estimated coefficients have to be compared to the zero coefficient of the reference category. Alternative categorization setups are given by omitting the constant, the sum coding (restrict the coefficients to sum up to 0), and the Helmert coding.

Example 10 (Credit default on categorized AGE)

We have chosen the intervals (18, 23], (23, 28], ..., (68, 75] as categories. Except for the last interval all of them are of the same length. The first interval (18, 23] is chosen for the reference such that we will estimate coefficients only for the remaining 10 intervals.

Frequency barplots for the intervals and estimated default probabilities are displayed in Fig. 7.4. The resulting coefficients for this model are listed in Table 7.7. We see here that all coefficient estimates are negative. This means, keeping in mind that the group of youngest credit applicants is the reference, that all applicants from other age groups have an (estimated) lower default probability. However, we do not have a true decrease in the default probabilities with AGE since the coefficients do not form a decreasing sequence. In the range from age 33 to 63 we find two local minima and maxima for the estimated default probabilities.

Table 7.7. Credit default on categorized AGE (logit model)

Variable	Coefficients	<i>t</i> -values
constant	-0.4055	-2.036
AGE (23,28]	-0.2029	-0.836
AGE (28,33]	-0.3292	-1.294
AGE (33,38]	-0.9144	-3.320
AGE (38,43]	-0.5447	-1.842
AGE (43,48]	-0.6763	-2.072
AGE (48,53]	-0.8076	-2.035
AGE (53,58]	-0.5108	-1.206
AGE (58,63]	-0.4055	-0.864
AGE (63,68]	-0.7577	-1.379
AGE (68,75]	-1.3863	-1.263
Deviance	1203.2	
df	989	
AIC	1225.2	
Iterations	4	

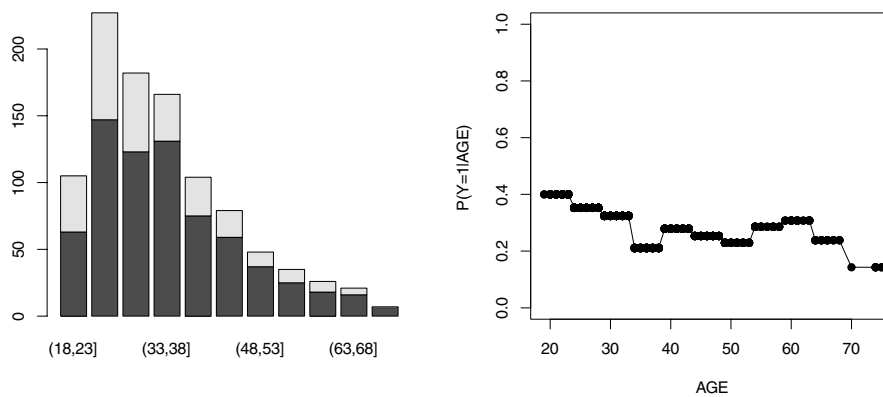


Figure 7.4. Credit default on categorized AGE, *left*: frequency barplots of categorized AGE for $Y = 1$ (*light*) and $Y = 0$ (*dark*), *right*: estimated probabilities

It is interesting to note that the deviance of the categorized AGE fit is the smallest that we obtained up to now. This is explained by the fact that we have fitted the most flexible model here. Unfortunately, this flexibility pays with the number of parameters. The AIC criterion as a compromise between goodness-of-fit and number of parameters states that all previous fitted models are preferable. Nevertheless, categorization is a valuable tool to explore if there are nonlinear effects. A related technique is local regression smoothing which is shortly reviewed in Sect. 7.5.8.

The estimation of default probabilities and the prediction of credit default should incorporate more than only one explanatory variable. Before fitting the full model with all available information, we discuss the modeling of interaction effects.

Example 11 (Credit default on AGE and AMOUNT)

The variable AMOUNT is the second continuous explanatory variable in the credit data set. (Recall that duration is quantitative as well but quasi-discrete.) We will therefore use AGE and AMOUNT to illustrate the effects of the simultaneous use of two explanatory variables. A very simple model is of course $\text{logit}\{P(Y = 1|AGE,AMOUNT)\} = \beta_0 + \beta_1 AGE + \beta_2 AMOUNT$. This model, however, separates the impact of AGE and AMOUNT into additive components. The effect of having both characteristics simultaneously is modeled by adding the multiplicative interaction term AGE*AMOUNT. On the other hand we have seen that at least AGE should be complemented by a quadratic term. For that reason we compare the linear interaction model $\text{logit}\{P(Y = 1|AGE,AMOUNT)\} = \beta_0 + \beta_1 AGE + \beta_2 AMOUNT + \beta_3 AGE * AMOUNT$ with a specification using quadratic terms and a third model specification using both, quadratic and interaction terms.

Table 7.8 shows the results for all three fitted models. The model with quadratic and interaction terms has the smallest AIC of the three fits. Pairwise LR tests show,

Table 7.8. Credit default on AGE and AMOUNT (logit model)

Variable	Coefficient	t-value	Coefficient	t-value	Coefficient	t-value
constant	0.0159	-0.044	1.1815	1.668	1.4864	2.011
AGE	-0.0350	-3.465	-0.1012	-2.768	-0.1083	-2.916
AGE**2	-	-	9.86×10^{-4}	2.251	9.32×10^{-4}	2.100
AMOUNT	-2.80×10^{-5}	-0.365	-7.29×10^{-6}	-0.098	-1.18×10^{-4}	-1.118
AMOUNT**2	-	-	1.05×10^{-8}	1.753	9.51×10^{-9}	1.594
AGE*AMOUNT	3.99×10^{-6}	1.951	-	-	3.37×10^{-6}	1.553
Deviance	1185.1		1180.2		1177.7	
df	996		995		994	
AIC	1193.1		1190.2		1189.7	
Iterations	4		4		4	

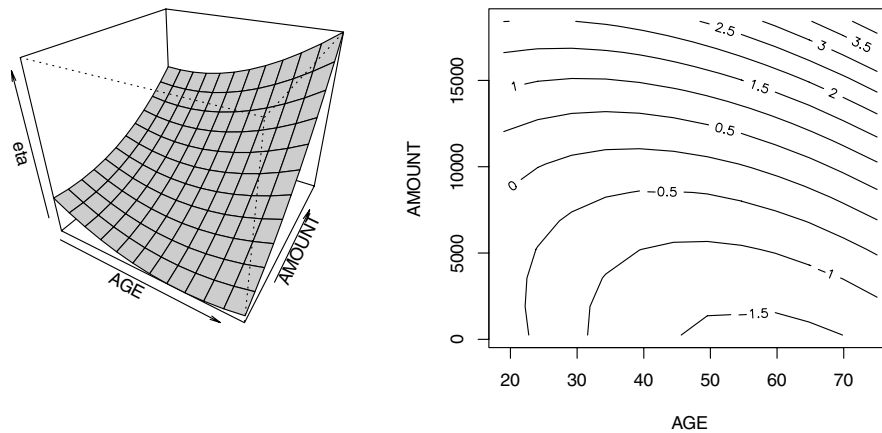


Figure 7.5. Credit default on AGE and AMOUNT using quadratic and interaction terms, *left*: surface and *right*: contours of the fitted η function

however, that the largest of the three models is not significantly better than the model without the interaction term. The obtained surface on AGE and AMOUNT from the quadratic+interaction fit is displayed in Fig. 7.5.

Let us remark that interaction terms can also be defined for categorical variables. In this case interaction is modeled by including dummy variables for all possible combinations of categories. This may largely increase the number of parameters to estimate.

12

Example 12 (Credit default on the full set of explanatory variables)

In a final analysis we present now the results for the full set of variables from Table 7.3. We first estimated a logit model using all variables (AGE and AMOUNT also with quadratic and interaction terms). Most of the estimated coefficients in the second column of Table 7.9 have the expected sign. For example, the default probability decreases if previous loan were paid back without problems, the credit applicant is employed and has some savings, and the loan is used to buy a car (rather than to invest the loan into goods which cannot serve as a security). A bit surprising is the fact that house owners seem to have higher default probabilities. This might be explained by the fact that house owners usually have additional obligations. The DURATION variable is categorized as described above. Again we have used the first category (loans up to 9 months) as reference. Since the series of DURATION coefficients is monotone increasing, we can conclude that longer duration increases the default probability. This is also plausible.

After fitting the full model we have run an automatic stepwise model selection based on AIC. This reveals that the insignificant terms AGE*AMOUNT and EMPLOYED should be omitted. The fitted coefficients for this final model are displayed in the fourth column of Table 7.9.

Table 7.9. Credit default on full set of variables (logit model)

Variable	Coefficient	<i>t</i> -value	Coefficient	<i>t</i> -value
constant	1.3345	1.592	0.8992	1.161
AGE	-0.0942	-2.359	-0.0942	-2.397
AGE**2	8.33×10^{-4}	1.741	9.35×10^{-4}	1.991
AMOUNT	-2.51×10^{-4}	-1.966	-1.67×10^{-4}	-1.705
AMOUNT**2	1.73×10^{-8}	2.370	1.77×10^{-8}	2.429
AGE*AMOUNT	2.36×10^{-6}	1.010	-	-
PREVIOUS	-0.7633	-4.652	-0.7775	-4.652
EMPLOYED	-0.3104	-1.015	-	-
DURATION (9, 12]	0.5658	1.978	0.5633	1.976
DURATION (12, 18]	0.8979	3.067	0.9127	3.126
DURATION (18, 24]	0.9812	3.346	0.9673	3.308
DURATION ≥ 24	1.5501	4.768	1.5258	4.710
SAVINGS	-0.9836	-4.402	-0.9778	-4.388
PURPOSE	-0.3629	-2.092	-0.3557	-2.051
HOUSE	0.6603	3.155	0.7014	3.396
Deviance	1091.5		1093.5	
df	985		987	
AIC	1121.5		1119.5	
Iterations	4		4	

Complements and Extensions

7.5

For further reading on GLM we refer to the textbooks of Dobson, 2001, McCullagh and Nelder, 1989 and Hardin and Hilbe, 2001 (the latter with a special focus on STATA). Venables and Ripley, 2002, Chap. 7 and Gill, 2000 present the topic of generalized linear models in a very compact form. Collett, 1991, Agresti, 1996, Cox and Snell, 1989, and Bishop et al., 1975 are standard references for analyzing categorical responses. We recommend the monographs of Fahrmeir and Tutz, 1994 and Lindsey, 1997 for a detailed introduction to GLM with a focus on multivariate, longitudinal and spatial data. In the following sections we will shortly review some specific variants and enhancements of the GLM.

Weighted Regression

7.5.1

Prior weights can be incorporated to the generalized linear model by considering the exponential density in the form

$$f(y_i, \theta_i, \psi) = \exp \left[\frac{w_i \{y\theta - b(\theta)\}}{a(\psi)} + c(y, \psi, w_i) \right].$$

This requires to optimize the sample log-likelihood

$$\ell(\mathbf{Y}, \boldsymbol{\mu}, \psi) = \sum_{i=1}^n w_i \left\{ \frac{Y_i \theta_i - b(\theta_i)}{a(\psi)} - c(Y_i, \psi, w_i) \right\}$$

or its equivalent, the deviance.

The weights w_i can be 0 or 1 in the simplest case that one wants to exclude specific observations from the estimation. The typical case of applying weights is the case of repeated independent realizations.

7.5.2 Overdispersion

Overdispersion may occur in one-parameter exponential families where the variance is supposed to be a function of the mean. This concerns in particular the binomial or Poisson families where we have $EY = \mu$ and $\text{Var}(Y) = \mu(1 - \mu/k)$ or $\text{Var}(Y) = \mu$, respectively. Overdispersion means that the actually observed variance from the data is larger than the variance imposed by the model. The source for this may be a lack of independence in the data or a misspecification of the model. One possible approach is to use alternative models that allows for a nuisance parameter in the variance, as an example think of the negative binomial instead of the Poisson distribution. For detailed discussions on overdispersion see Collett, 1991 and Agresti, 1990.

7.5.3 Quasi- or Pseudo-Likelihood

Let us remark that in the case that the distribution of Y itself is unknown but its two first moments can be specified, the quasi-likelihood function may replace the log-likelihood function. This means we still assume that

$$E(Y) = \mu,$$

$$\text{Var}(Y) = a(\psi) V(\mu).$$

The quasi-likelihood function is defined through

$$\ell(y, \theta, \psi) = \frac{1}{a(\psi)} \int_{\mu(\theta)}^y \frac{(s-y)}{V(s)} ds, \quad (7.18)$$

see Nelder and Wedderburn, 1972. If Y comes from an exponential family then the derivatives of the log-likelihood and quasi-likelihood function coincide. Thus, (7.18) establishes in fact a generalization of the likelihood approach.

7.5.4 Multinomial Responses

A multinomial model (or nominal logistic regression) is applied if the response for each observation i is one out of more than two alternatives (categories). For

identification one of the categories has to be chosen as reference category; without loss of generality we use here the first category. Denote by π_j the probability $P(Y = j|\mathbf{X})$, then we can consider the logits with respect to the first category, i.e.

$$\text{logit}(\pi_j) = \log\left(\frac{\pi_j}{\pi_1}\right) = \mathbf{X}_j^\top \boldsymbol{\beta}_j.$$

The terms \mathbf{X}_j and $\boldsymbol{\beta}_j$ indicate that the explanatory variables and their corresponding coefficients may depend on category j . Equivalently we can define the model by

$$P(Y = 1|\mathbf{X}) = \frac{1}{1 + \sum_{k=2}^J \exp(\mathbf{X}_k^\top \boldsymbol{\beta}_k)}$$

$$P(Y = j|\mathbf{X}) = \frac{\mathbf{X}_j^\top \boldsymbol{\beta}_j}{1 + \sum_{k=2}^J \exp(\mathbf{X}_k^\top \boldsymbol{\beta}_k)}.$$

It is easy to recognize that the logit model is a special case of the multinomial model for exactly two alternatives.

If the categories are ordered in some natural way then this additional information can be taken into account. A latent variable approach leads to the cumulative logit model or the ordered probit model. We refer here to Dobson, 2001, Sect. 8.4 and Greene, 2000, Chap. 21 for ordinal logistic regression and ordered probit analysis, respectively.

Contingency Tables

The simplest form of a contingency table with one factor and a predetermined

Category	1	2	...	J	Σ
Frequency	Y_1	Y_2	...	Y_J	n

sample size n of observations is appropriately described by a multinomial distribution and can hence be fitted by the multinomial logit model introduced in Sect. 7.5.4. We could be for instance be interested in comparing the trivial model $EY_1 = \dots = EY_J = \mu$ to the model $EY_2 = \mu_2, \dots, EY_J = \mu_J$ (again we use the first category as reference). As before further explanatory variables can be included into the model.

Two-way or higher dimensional contingency tables involve a large variety of possible models. Let explain this with the help of the following two-way setup: Here we assume to have two factors, one with realizations $1, \dots, J$, the other with realizations $1, \dots, K$. If the Y_{jk} are independent Poisson variables with parameters μ_{jk} , then their sum is a Poisson variable with parameter $E(n) = \mu = \sum \mu_{jk}$. The Poisson assumption implies that the number of observations n is a random variable. Conditional on n , the joint distribution of the Y_{jk} is the multinomial distribution.

Category	1	2	...	J	Σ
1	Y_{11}	Y_{12}	...	Y_{1J}	$n_{1\bullet}$
2	Y_{21}	Y_{22}	...	Y_{2J}	$n_{2\bullet}$
\vdots	\vdots	\vdots	\ddots	\vdots	\vdots
K	Y_{K1}	Y_{K2}	...	Y_{KJ}	$n_{K\bullet}$
Σ	$n_{\bullet 1}$	$n_{\bullet 2}$...	$n_{\bullet J}$	n

Without additional explanatory variables, one is typically interested in estimating models of the type

$$\log(EY_{jk}) = \beta_0 + \beta_j + \beta_k$$

in order to compare this with the saturated model $\log(EY_{jk}) = \beta_0 + \beta_j + \beta_k + \beta_{jk}$. If the former model holds then the two factors are independent. Another hypothetical model could be of the form $\log(EY_{jk}) = \beta_0 + \beta_j$ to check whether the second factor matters at all. As in the multinomial case, further explanatory variables can be included. This type of models is consequently termed log-linear model. For more details see for example Dobson, 2001, Chap. 9 and McCullagh and Nelder, 1989, Chap. 6.

7.5.6 Survival Analysis

Survival data are characterized by non-negative observations which typically have a skewed distribution. An additional complication arises due to the fact that the observation period may end before the individual fails such that censored data may occur. The exponential distribution with density $f(y, \theta) = \theta e^{-\theta y}$ is a very simple example for a survival distribution. In this special case the survivor function (the probability to survive beyond y) is given by $S(y) = e^{-\theta y}$ and the hazard function (the probability of death within y and $y+dy$ after survival up to y) equals $h(y, \theta) = \theta$. Given additional explanatory variables this function is typically modeled by

$$h(y, \theta) = \exp(\mathbf{X}^\top \boldsymbol{\beta}).$$

Extensions of this model are given by using the Weibull distribution leading to non-constant hazards and Cox' proportional hazards model (Cox, 1972) which uses a semiparametric approach. More material on survival analysis can be found in Chap. III.12.

7.5.7 Clustered Data

Clustered data in relation to regression models mean that data from known groups ("clusters") are observed. Often these are the result of repeated measurements on the same individuals at different time points. For example, imagine the analysis of the effect of a medical treatment on patients or the repeated surveying of

households in socio-economic panel studies. Here, all observations on the same individual form a cluster. We speak of longitudinal or panel data in that case. The latter term is typically used in the econometric literature.

When using clustered data we have to take into account that observations from the same cluster are correlated. Using a model designed for independent data may lead to biased results or at least significantly reduce the efficiency of the estimates.

A simple individual model equation could be written as follows:

$$E(Y_{ij}|\mathbf{X}_{ij}) = G^{-1}(\mathbf{X}_{ij}^T \boldsymbol{\beta}_j).$$

Here i is used to denote the i th individual observation in the j th cluster. Of course more complex specifications, for example with hierarchical clusters, can be formulated as well.

There is a waste amount of literature which deals with many different possible model specifications. A comprehensive resource for linear and nonlinear mixed effect models (LME, NLME) for continuous responses is Pinheiro and Bates, 2000. The term “mixed” here refers to the fact that these models include additional random and/or fixed effect components to allow for correlation within and heterogeneity between the clusters.

For generalized linear mixed models (GLMM), i.e. clustered observations with responses from GLM-type distribution, several approaches are possible. For repeated observations, Liang and Zeger, 1986 and Zeger and Liang, 1986 propose to use generalized estimating equations (GEE) which result in a quasi-likelihood estimator. They show that the correlation matrix of \mathbf{Y}_j , the response observations from one cluster, can be replaced by a “working correlation” as long as the moments of \mathbf{Y}_j are correctly specified. Useful working correlations depend on a small number of parameters. For longitudinal data an autoregressive working correlation can be used for example. For more details on GEE see also the monograph by Diggle et al., 2002. In the econometric literature longitudinal or panel data are analyzed with a focus on continuous and binary responses. Standard references for econometric panel data analyses are Hsiao, 1990 and Arellano, 2003. Models for clustered data with complex hierarchical structure are often denoted as multilevel models. We refer to the monograph of Goldstein, 2003 for an overview.

Semiparametric Generalized Linear Models

 7.5.8

Nonparametric components can be incorporated into the GLM at different places. For example, it is possible to estimate a single index model

$$E(Y|\mathbf{X}) = g(\mathbf{X}^T \boldsymbol{\beta})$$

which differs from the GLM by its unknown smooth link function $g(\bullet)$. The parameter vector $\boldsymbol{\beta}$ in this model can then be only identified up to scale. The estimation of such models has been studied e.g. by Ichimura, 1993, Weisberg and Welsh, 1994 and Gallant and Nychka, 1987.

Local regression in combination with likelihood-based estimation is introduced in Loader, 1999. This concerns models of the form

$$E(Y|\mathbf{X}) = G^{-1} \{m(\mathbf{X})\},$$

where m is an unknown smooth (possibly multidimensional) function. Further examples of semiparametric GLM are generalized additive and generalized partial linear models (GAM, GPLM). These models are able to handle (additional) non-parametric components in the function η . For example, the GAM is specified in this simplest form by

$$E(Y|\mathbf{X}) = G^{-1} \left\{ \beta_0 + \sum_{j=1}^p m_j(X_j) \right\}.$$

Here the m_j denote univariate (or low dimensional) unknown smooth functions which have to be estimated. For their identification it should be assumed, that $E m(X_j) = 0$. The generalized partial linear model combines a linear and a non-parametric function in the function η and is specified as

$$E(Y|\mathbf{X}) = G^{-1} \{ \mathbf{X}_1^\top \boldsymbol{\beta} + m(\mathbf{X}_2) \}.$$

13

Example 13 (Semiparametric credit model)

We have fitted a generalized partial linear model as a variant of the final model from Example 12. The continuous variables AGE and AMOUNT were used as arguments for the nonparametric component. All other variables of the final model have been included to the linear part of the index function η . Figure 7.6 shows the

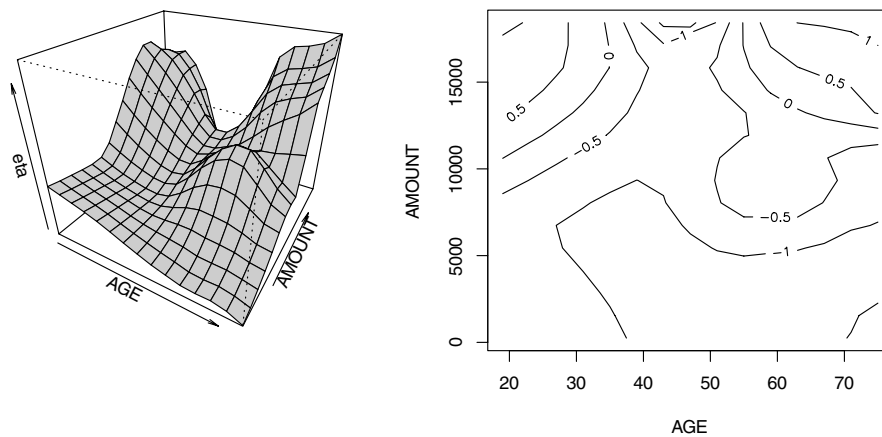


Figure 7.6. Credit default on AGE and AMOUNT using a nonparametric function, left: surface and right: contours of the fitted function on AGE and AMOUNT

estimated nonparametric function of AGE and AMOUNT. Although the stepwise model selection in Example 12 indicated that there is no interaction between AGE and AMOUNT, we see now that this interaction could be in fact of some more sophisticated form. The estimation was performed using a generalization of the Speckman, 1988 estimator to generalized models. The local kernel weights are calculated from a Quartic (Biweight) kernel function using bandwidths approximately equal to 33.3% of the ranges of AGE and AMOUNT, respectively. Details on the used kernel based estimation can be found in Severini and Staniswalis, 1994 and Müller, 2001.

Some more material on semiparametric regression can be found in Chaps. III.5 and III.10 of this handbook. For a detailed introduction to semiparametric extensions of GLM we refer to the textbooks by Hastie and Tibshirani, 1990, Härdle et al., 2004, Ruppert et al., 2003, and Green and Silverman, 1994.

References

- Agresti, A. (1990). *Categorical Data Analysis*. Wiley, New York.
- Agresti, A. (1996). *An Introduction to Categorical Data Analysis*. Wiley, New York.
- Akaike, H. (1973). Information theory and an extension of the maximum likelihood principle. In Petrov, B. N. and Csàdki, F., editors, *Second International Symposium on Information Theory*, pages 267–281. Akademiai Kiadó, Budapest.
- Arellano, M. (2003). *Panel Data Econometrics*. Oxford University Press.
- Bishop, Y. M. M., Fienberg, S. E., and Holland, P. W. (1975). *Discrete Multivariate Analysis: Theory and Practice*. MIT Press, Cambridge.
- Box, G. and Cox, D. (1964). An analysis of transformations. *Journal of the Royal Statistical Society, Series B*, 26:211–243.
- Collett, D. (1991). *Modelling Binary Data*. Chapman and Hall, London.
- Cox, D. R. (1972). Regression models and life tables (with discussion). *Journal of the Royal Statistical Society, Series B*, 74:187–220.
- Cox, D. R. and Snell, E. J. (1989). *Analysis of Binary Data*, volume 32 of *Monographs on Statistics and Applied Probability*. Chapman and Hall, London, second edition.
- Diggle, P., Heagerty, P., Liang, K.-L., and Zeger, S. (2002). *Analysis of Longitudinal Data*. Oxford University Press, second edition.
- Dobson, A. J. (2001). *An Introduction to Generalized Linear Models*. Chapman and Hall, London, second edition.
- Fahrmeir, L. and Kaufmann, H. (1984). Consistency and asymptotic normality of the maximum-likelihood estimator in generalized linear models. *Annals of Statistics*, 13:342–368.

- Fahrmeir, L. and Tutz, G. (1994). *Multivariate Statistical Modelling Based on Generalized Linear Models*. Springer, Heidelberg.
- Gallant, A. and Nychka, D. (1987). Semi-nonparametric maximum likelihood estimation. *Econometrica*, 55(2):363–390.
- Gill, J. (2000). *Generalized Linear Models: A Unified Approach*. Sage University Paper Series on Quantitative Applications in the Social Sciences, 07-134, Thousand Oaks, CA.
- Goldstein, H. (2003). *Multilevel Statistical Models*. Hodder Arnold, London.
- Green, P. J. and Silverman, B. W. (1994). *Nonparametric Regression and Generalized Linear Models*, volume 58 of *Monographs on Statistics and Applied Probability*. Chapman and Hall, London.
- Greene, W. H. (2000). *Econometric Analysis*. Prentice Hall, Upper Saddle River, New Jersey, 4th edition.
- Hardin, J. and Hilbe, J. (2001). *Generalized Linear Models and Extensions*. Stata Press.
- Härdle, W., Müller, M., Sperlich, S., and Werwatz, A. (2004). *Nonparametric and Semiparametric Modeling: An Introduction*. Springer, New York.
- Hastie, T. J. and Tibshirani, R. J. (1990). *Generalized Additive Models*, volume 43 of *Monographs on Statistics and Applied Probability*. Chapman and Hall, London.
- Hsiao, C. (1990). *Analysis of Panel Data*. Econometric Society Monographs No. 11. Cambridge University Press.
- Ichimura, H. (1993). Semiparametric least squares (SLS) and weighted SLS estimation of single-index models. *Journal of Econometrics*, 58:71–120.
- Liang, K. Y. and Zeger, S. L. (1986). Longitudinal data analysis using generalized linear models. *Biometrika*, 73:13–22.
- Lindsey, J. K. (1997). *Applying Generalized Linear Models*. Springer, New York.
- Loader, C. (1999). *Local Regression and Likelihood*. Springer, New York.
- McCullagh, P. and Nelder, J. A. (1989). *Generalized Linear Models*, volume 37 of *Monographs on Statistics and Applied Probability*. Chapman and Hall, London, 2 edition.
- Müller, M. (2001). Estimation and testing in generalized partial linear models – a comparative study. *Statistics and Computing*, 11:299–309.
- Nelder, J. A. and Wedderburn, R. W. M. (1972). Generalized linear models. *Journal of the Royal Statistical Society, Series A*, 135(3):370–384.
- Pinheiro, J. C. and Bates, D. M. (2000). *Mixed-Effects Models in S and S-PLUS*. Springer, New York.
- Ruppert, D., Wand, M. P., and Carroll, R. J. (1990). *Semiparametric Regression*. Cambridge University Press.
- Schwarz, G. (1978). Estimating the dimension of a model. *Annals of Statistics*, 6:461–464.
- Severini, T. A. and Staniswalis, J. G. (1994). Quasi-likelihood estimation in semi-parametric models. *Journal of the American Statistical Association*, 89:501–511.
- Speckman, P. E. (1988). Regression analysis for partially linear models. *Journal of the Royal Statistical Society, Series B*, 50:413–436.

- Turlach, B. A. (1994). Computer-aided additive modeling. Doctoral Thesis, Université Catholique de Louvain, Belgium.
- Venables, W. N. and Ripley, B. (2002). *Modern Applied Statistics with S*. Springer, New York, 4th edition.
- Weisberg, S. and Welsh, A. H. (1994). Adapting for the missing link. *Annals of Statistics*, 22:1674–1700.
- Zeger, S. L. and Liang, K. Y. a. (1986). Longitudinal data analysis for discrete and continuous outcomes. *Biometrics*, 42:121–130.

(Non) Linear Regression Modeling

III.8

Pavel Čížek

8.1	<i>Linear Regression Modeling</i>	622
	Fitting of Linear Regression	623
	Multicollinearity	625
	Variable Selection	627
	Principle Components Regression	632
	Shrinkage Estimators	634
	Ridge Regression	634
	Continuum Regression	638
	Lasso	639
	Partial Least Squares	641
	Comparison of the Methods	643
8.2	<i>Nonlinear Regression Modeling</i>	644
	Fitting of Nonlinear Regression	645
	Statistical Inference	647
	Ill-conditioned Nonlinear System	648

We will study causal relationships of a known form between random variables. Given a model, we distinguish one or more dependent (endogenous) variables $Y = (Y_1, \dots, Y_l)$, $l \in \mathbb{N}$, which are explained by a model, and independent (exogenous, explanatory) variables $X = (X_1, \dots, X_p)$, $p \in \mathbb{N}$, which explain or predict the dependent variables by means of the model. Such relationships and models are commonly referred to as regression models.

A regression model describes the relationship between the dependent and independent variables. In this chapter, we restrict our attention to models with a form known up to a finite number of unspecified parameters. The model can be either linear in parameters,

$$Y = X^\top \beta_0 + \varepsilon,$$

or nonlinear,

$$Y = h(X, \beta_0) + \varepsilon,$$

where β represents a vector or a matrix of unknown parameters, ε is the error term (fluctuations caused by unobservable quantities), and h is a known regression function. The unknown parameters β are to be estimated from observed realizations $\{y_{1i}, \dots, y_{li}\}_{i=1}^n$ and $\{x_{1i}, \dots, x_{pi}\}_{i=1}^n$ of random variables Y and X .

Here we discuss both kinds of models, primarily from the least-squares estimation point of view, in Sects. 8.1 and 8.2, respectively. Both sections present the main facts concerning the fitting of these models and relevant inference, whereby their focus is above all on the estimation of these regression models under near and exact multicollinearity.

8.1 Linear Regression Modeling

Let us first study the linear regression model $Y = X^\top \beta_0 + \varepsilon$ assuming $E(\varepsilon|X) = 0$. Unless said otherwise, we consider here only one dependent variable Y . The unknown vector $\beta^0 = (\beta_1^0, \dots, \beta_p^0)$ is to be estimated given observations $y = (y_1, \dots, y_n) \in \mathbb{R}^n$ and $\{x_i\}_{i=1}^n = \{(x_{1i}, \dots, x_{pi})\}_{i=1}^n$ of random variables Y and X ; let us denote $X = (x_1, \dots, x_n)^\top \in \mathbb{R}^{n \times p}$ and let x_k be the k th column of X . Thus, the linear regression model can be written in terms of observations as

$$y = X\beta_0 + \varepsilon = \sum_{k=1}^p x_k \beta_k^0 + \varepsilon, \quad (8.1)$$

where $\varepsilon = (\varepsilon_1, \dots, \varepsilon_n) \in \mathbb{R}^n$.

Section 8.1.1 summarizes how to estimate the model (8.1) by the method of least squares. Later, we specify what ill-conditioning and multicollinearity are in Sect. 8.1.2 and discuss methods dealing with it in Sects. 8.1.3–8.1.9.

Fitting of Linear Regression

8.1.1

Let us first review the least squares estimation and its main properties to facilitate easier understanding of the fitting procedures discussed further. For a detailed overview of linear regression modeling see Rao and Toutenberg (1999).

The *least squares* (LS) approach to the estimation of (8.1) searches an estimate $\widehat{\boldsymbol{\beta}}$ of unknown parameters $\boldsymbol{\beta}_0$ by minimizing the sum of squared differences between the observed values y_i and the predicted ones $\widehat{y}_i(\boldsymbol{\beta}) = \mathbf{x}_i^\top \boldsymbol{\beta}$.

Definition 1 The least squares estimate of linear regression model (8.1) is defined by

$$\widehat{\boldsymbol{\beta}}^{\text{LS}} = \underset{\boldsymbol{\beta} \in \mathbb{R}^p}{\operatorname{argmin}} \sum_{i=1}^n \{y_i - \widehat{y}_i(\boldsymbol{\beta})\}^2 = \underset{\boldsymbol{\beta} \in \mathbb{R}^p}{\operatorname{argmin}} \sum_{i=1}^n (y_i - \mathbf{x}_i^\top \boldsymbol{\beta})^2. \quad (8.2)$$

This differentiable problem can be expressed as minimization of

$$(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})^\top (\mathbf{y} - \mathbf{X}\boldsymbol{\beta}) = \mathbf{y}^\top \mathbf{y} - 2\boldsymbol{\beta}^\top \mathbf{X}^\top \mathbf{y} + \boldsymbol{\beta}^\top \mathbf{X}^\top \mathbf{X} \boldsymbol{\beta}$$

with respect to $\boldsymbol{\beta}$ and the corresponding first-order conditions are

$$-\mathbf{X}^\top \mathbf{y} + \mathbf{X}^\top \mathbf{X} \boldsymbol{\beta} = 0 \implies \mathbf{X}^\top \mathbf{X} \boldsymbol{\beta} = \mathbf{X}^\top \mathbf{y}. \quad (8.3)$$

They are commonly referred to as normal equations and identify the global minimum of (8.2) as long as the second order conditions $\mathbf{X}^\top \mathbf{X} > 0$ hold; that is, the matrix $\mathbf{X}^\top \mathbf{X}$ is supposed to be positive definite, or equivalently, non-singular. (This mirrors an often used assumption specified in terms of the underlying random variable X : $E(\mathbf{X}\mathbf{X}^\top) > 0$.) Provided that $\mathbf{X}^\top \mathbf{X} > 0$ and $E(\boldsymbol{\varepsilon}|\mathbf{X}) = 0$, the LS estimator is unbiased and can be found as a solution of (8.3)

$$\widehat{\boldsymbol{\beta}}^{\text{LS}} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y}. \quad (8.4)$$

Additionally, it is the best linear unbiased estimator of (8.1), see Amemiya (1985).

Theorem 1 (Gauss–Markov)

Assume that $E(\boldsymbol{\varepsilon}|\mathbf{X}) = 0$, $E(\boldsymbol{\varepsilon}^2|\mathbf{X}) = \sigma^2 \mathbf{I}_n$, and $\mathbf{X}^\top \mathbf{X}$ is non-singular. Let $\widehat{\boldsymbol{\beta}} = \mathbf{C}^\top \mathbf{y}$, where \mathbf{C} is a $t \times p$ matrix orthogonal to \mathbf{X} , $\mathbf{C}^\top \mathbf{X} = \mathbf{I}$. Then $\operatorname{Var}(\widehat{\boldsymbol{\beta}}) - \operatorname{Var}(\widehat{\boldsymbol{\beta}}^{\text{LS}}) > 0$ is a positive definite matrix for any $\widehat{\boldsymbol{\beta}} \neq \widehat{\boldsymbol{\beta}}^{\text{LS}}$.

Finally, the LS estimate actually coincides with the maximum likelihood estimates provided that random errors $\boldsymbol{\varepsilon}$ are normally distributed (in addition to the assumptions of Theorem 1) and shares then the asymptotic properties of the maximum likelihood estimation (see Amemiya, 1985).

Computing LS Estimates

The LS estimate $\widehat{\boldsymbol{\beta}}^{\text{LS}}$ can be and often is found by directly solving the system of linear equations (8.3) or evaluating formula (8.4), which involves a matrix inversion. Both

direct and iterative methods for solving systems of linear equations are presented in Chap. II.4. Although this straightforward computation may work well for many regression problems, it often leads to an unnecessary loss of precision, see Miller (2002). Additionally, it is not very suitable if the matrix $X^T X$ is ill-conditioned (a regression problem is called ill-conditioned if a small change in data causes large changes in estimates) or nearly singular (multicollinearity) because it is not numerically stable. Being concerned mainly about statistical consequences of multicollinearity, the numerical issues regarding the identification and treatment of ill-conditioned regression models are beyond the scope of this contribution. Let us refer an interested reader Barlow (1993), Björck (1996), Miller (2002), Thisted (1988) and Wang et al. (1990).

Let us now briefly review a class of numerically more stable algorithms for the LS minimization. They are based on orthogonal transformations. Assuming a matrix $Q \in \mathbb{R}^{n \times n}$ is an orthonormal matrix, $Q^T Q = Q Q^T = I_n$,

$$(\mathbf{y} - X\boldsymbol{\beta})^T (\mathbf{y} - X\boldsymbol{\beta}) = (Q\mathbf{y} - QX\boldsymbol{\beta})^T (Q\mathbf{y} - QX\boldsymbol{\beta}) .$$

Thus, multiplying a regression model by an orthonormal matrix does not change it from the LS point of view. Since every matrix X can be decomposed into the product $Q_x R_x$ (the QR decomposition), where Q_x is an orthonormal matrix and R_x is an upper triangular matrix, pre-multiplying (8.1) by Q_x^T produces

$$Q_x^T \mathbf{y} = R_x \boldsymbol{\beta} + Q_x^T \boldsymbol{\varepsilon} , \quad (8.5)$$

where $R_x = (R_1, R_2)^T$, $R_1 \in \mathbb{R}^{p \times p}$ is an upper triangular matrix, and $R_2 \in \mathbb{R}^{(n-p) \times p}$ is a zero matrix. Hence, the sum of squares to minimize can be written as

$$(Q_x^T \mathbf{y} - R_x \boldsymbol{\beta})^T (Q_x^T \mathbf{y} - R_x \boldsymbol{\beta}) = (\mathbf{y}_1 - R_1 \boldsymbol{\beta})^T (\mathbf{y}_1 - R_1 \boldsymbol{\beta}) + \mathbf{y}_2^T \mathbf{y}_2 ,$$

where $\mathbf{y}_1 \in \mathbb{R}^p$ and $\mathbf{y}_2 \in \mathbb{R}^{n-p}$ form $Q_x^T \mathbf{y} = (\mathbf{y}_1^T, \mathbf{y}_2^T)^T$. The LS estimate is then obtained from the upper triangular system $R_1 \boldsymbol{\beta} = \mathbf{y}_1$, which is trivial to solve by backward substitution. There are many algorithms for constructing a suitable QR decomposition for finding LS estimates, such as the Householder or Givens transformations; see Chap. II.4, more details.

LS Inference

Linear regression modeling does not naturally consist only of obtaining a point estimate $\hat{\boldsymbol{\beta}}^{\text{LS}}$. One needs to measure the variance of the estimates in order to construct confidence intervals or test hypotheses. Additionally, one should assess the quality of the regression fit. Most such measures are based on regression residuals $\mathbf{e} = \mathbf{y} - X\hat{\boldsymbol{\beta}}$. We briefly review the most important regression statistics, and next, indicate how it is possible to compute them if the LS regression is estimated by means of some orthogonalization procedure described in the previous paragraph.

The most important measures used in statistics to assess model fit and inference are the total sum of squares $TSS = (\mathbf{y} - \bar{y})^T (\mathbf{y} - \bar{y}) = \sum_{i=1}^n (y_i - \bar{y})^2$, where $\bar{y} = \sum_{i=1}^n y_i / n$, the residual sum of squares $RSS = \mathbf{e}^T \mathbf{e} = \sum_{i=1}^n e_i^2$, and the comple-

mentary regression sum of squares $RegSS = (\mathbf{y} - \hat{\mathbf{y}})^\top (\mathbf{y} - \hat{\mathbf{y}}) = \sum_{i=1}^n (y_i - \hat{y}_i)^2 = TSS - RSS$. Using these quantities, the regression fit can be evaluated; for example, the coefficient of determination $R^2 = 1 - RSS/TSS$ as well as many information criteria (modified \bar{R}^2 , Mallows and Akaike criteria, etc.; see Sect. 8.1.3). Additionally, they can be used to compute the variance of the estimates in simple cases. The variance of the estimates can be estimated by

$$\text{Var}(\hat{\boldsymbol{\beta}}^{LS}) = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{S}^{-1} \mathbf{X} (\mathbf{X}^\top \mathbf{X})^{-1}, \quad (8.6)$$

where \mathbf{S} represents an estimate of the covariance matrix $\text{Var}(\boldsymbol{\epsilon}) = \boldsymbol{\Sigma}$. Provided that the model is homoscedastic, $\boldsymbol{\Sigma} = \sigma^2 \mathbf{I}_n$, the residual variance σ^2 can be estimated as an average of squared residuals $s^2 = \mathbf{e}^\top \mathbf{e} / n$. Apart from the residual variance, one needs also an inverse of $(\mathbf{X}^\top \mathbf{X})^{-1}$, which will often be a by-product of solving normal equations.

Let us now describe how one computes these quantities if a numerically stable procedure based on the orthonormalization of normal equations is used. Let us assume we already constructed a QR decomposition of $\mathbf{X} = \mathbf{Q}_x \mathbf{R}_x$. Thus, $\mathbf{Q}_x \mathbf{Q}_x^\top = \mathbf{I}$ and $\mathbf{Q}_x^\top \mathbf{X} = \mathbf{R}_x$. RSS can be computed as

$$\begin{aligned} \text{RSS} &= \mathbf{e}^\top \mathbf{e} = (\mathbf{y} - \mathbf{X}\hat{\boldsymbol{\beta}})^\top (\mathbf{y} - \mathbf{X}\hat{\boldsymbol{\beta}}) = (\mathbf{y} - \mathbf{X}\hat{\boldsymbol{\beta}})^\top \mathbf{Q}_x \mathbf{Q}_x^\top (\mathbf{y} - \mathbf{X}\hat{\boldsymbol{\beta}}) \\ &= (\mathbf{Q}_x^\top \mathbf{y} - \mathbf{R}_x \mathbf{X}\hat{\boldsymbol{\beta}})^\top (\mathbf{Q}_x^\top \mathbf{y} - \mathbf{R}_x \mathbf{X}\hat{\boldsymbol{\beta}}). \end{aligned}$$

Consequently, RSS is invariant with respect to orthonormal transformations (8.5) of the regression model (8.1). The same conclusion applies also to TSS and $RegSS$, and consequently, to the variance estimation. Thus, it is possible to use the data in (8.5), transformed to achieve better numerical stability, for computing regression statistics of the original model (8.1).

Multicollinearity

8.1.2

Let us assume that the design matrix \mathbf{X} fixed. We talk about *multicollinearity* when there is a linear dependence among the variables in regression, that is, the columns of \mathbf{X} .

Definition 2 In model (8.1), the exact multicollinearity exists if there are real constants a_1, \dots, a_p such that $\sum_{k=1}^p |a_k| > 0$ and $\sum_{k=1}^p a_k \mathbf{x}_{\cdot k} = \mathbf{0}$.

2

The exact multicollinearity (also referred to as reduced-rank data) is relatively rare in linear regression models unless the number of explanatory variables is very large or even larger than the number of observations, $p \geq n$. This happens often in agriculture, chemometrics, sociology, and so on. For example, Miller (2002) uses data on the absorbances of infra-red rays at many different wavelength by chopped

meat, whereby the aim is to determine the moisture, fat, and protein content of the meat as a function of these absorbances. The study employs measurements at 100 wavelengths from 850 nm to 1050 nm, which gives rise to many possibly correlated variables.

When the number p of variables is small compared to the sample size n , near multicollinearity is more likely to occur: there are some real constants a_1, \dots, a_p such that $\sum_{k=1}^p |a_k| > 0$ and $\sum_{k=1}^p a_k \mathbf{x}_{.k} \approx 0$, where \approx denotes approximate equality. The multicollinearity in data does not have to arise only as a result of highly correlated variables (e.g., more measurements of the same characteristic by different sensors or methods), which by definition occurs in all applications where there are more variables than observations, but it could also result from the lack of information and variability in data.

Whereas the exact multicollinearity implies that $\mathbf{X}^T \mathbf{X}$ is singular and the LS estimator is not identified, the near multicollinearity permits non-singular matrix $\mathbf{X}^T \mathbf{X}$. The eigenvalues $\lambda_1 \leq \dots \leq \lambda_p$ of matrix $\mathbf{X}^T \mathbf{X}$ can give some indication concerning multicollinearity: if the smallest eigenvalue λ_1 equals zero, the matrix is singular and data are exactly multicollinear; if λ_1 is close to zero, near multicollinearity is present in data. Since measures based on eigenvalues depend on the parametrization of the model, they are not necessarily optimal and it is often easier to detect multicollinearity by looking at LS estimates and their behavior as discussed in next paragraph. See Björck (1996) and Leamer (1983) for more details on detection and treatment of ill-conditioned problems. (Nearly singular matrices are dealt with also in numerical mathematics. To measure near singularity, numerical mathematics uses conditioning numbers $d_k = \sqrt{\lambda_k/\lambda_1}$, which converge to infinity for singular matrices, that is, as $\lambda_1 \rightarrow 0$. Matrices with very large conditioning numbers are called ill-conditioned.)

The multicollinearity has important implications for LS. In the case of exact multicollinearity, matrix $\mathbf{X}^T \mathbf{X}$ does not have a full rank, hence the solution of the normal equations is not unique and the LS estimate $\hat{\boldsymbol{\beta}}^{\text{LS}}$ is not identified. One has to introduce additional restrictions to identify the LS estimate. On the other hand, even though near multicollinearity does not prevent the identification of LS, it negatively influences estimation results. Since both the estimate $\hat{\boldsymbol{\beta}}^{\text{LS}}$ and its variance are proportional to the inverse of $\mathbf{X}^T \mathbf{X}$, which is nearly singular under multicollinearity, near multicollinearity inflates $\hat{\boldsymbol{\beta}}^{\text{LS}}$, which may become unrealistically large, and variance $\text{Var}(\hat{\boldsymbol{\beta}}^{\text{LS}})$. Consequently, the corresponding t -statistics are typically very low. Moreover, due to the large values of $(\mathbf{X}^T \mathbf{X})^{-1}$, the least squares estimate $\hat{\boldsymbol{\beta}}^{\text{LS}} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y}$ reacts very sensitively to small changes in data. See Hocking (1996) and Montgomery et al. (2001) for a more detailed treatment and real-data examples of the effects of multicollinearity.

There are several strategies to limit adverse consequences of multicollinearity provided that one cannot improve the design of a model or experiment or get better data. First, one can impose an additional structure on the model. This strategy cannot be discussed in details since it is model specific, and in principle, it requires only to test a hypothesis concerning additional restrictions. Second, it

is possible to reduce the dimension of the space spanned by X , for example, by excluding some variables from the regression (Sects. 8.1.3 and 8.1.4). Third, one can also leave the class of unbiased estimators and try to find a biased estimator with a smaller variance and mean squared error. Assuming we want to judge the performance of an estimator $\hat{\beta}$ by its mean squared error (MSE), the motivation follows from

$$\begin{aligned} \text{MSE}(\hat{\beta}) &= E \left[(\hat{\beta} - \beta^0) (\hat{\beta} - \beta^0)^\top \right] \\ &= E \left[\{\hat{\beta} - E(\hat{\beta})\} \{\hat{\beta} - E(\hat{\beta})\}^\top \right] + [E\{E(\hat{\beta}) - \beta^0\}] [E\{E(\hat{\beta}) - \beta^0\}]^\top \\ &= \text{Var}(\hat{\beta}) + \text{Bias}(\hat{\beta}) \text{Bias}(\hat{\beta})^\top . \end{aligned}$$

Thus, it is possible that introducing a bias into estimation in such a way that the variance of estimates is significantly reduced can improve the estimator's MSE. There are many biased alternatives to the LS estimation as discussed in Sects. 8.1.5–8.1.9 and some of them even combine biased estimation with variable selection. In all cases, we present methods usable both in the case of near and exact multicollinearity.

Variable Selection

 8.1.3

The presence of multicollinearity may indicate that some explanatory variables are linear combinations of the other ones (note that this is more often a “feature” of data rather than of the model). Consequently, they do not improve explanatory power of a model and could be dropped from the model provided there is some justification for dropping them also on the model level rather than just dropping them to fix data problems. As a result of removing some variables, the matrix $X^\top X$ would not be (nearly) singular anymore.

Eliminating variables from a model is a special case of model selection procedures, which are discussed in details in Chap. III.1. Here we first discuss methods specific for variable selection within a single regression model, mainly variants of stepwise regression. Later, we deal with more general model selection methods, such as cross validation, that are useful both in the context of variable selection and of biased estimation discussed in Sects. 8.1.5–8.1.9. An overview and comparison of many classical variable selection is given, for example, in Miller (1984, 2002) and Montgomery et al. (2001). For discussion of computational issues related to model selection, see Kennedy and Gentle (1980) and Miller (2002).

Backward Elimination

A simple and often used method to eliminate non-significant variables from regression is *backward elimination*, a special case of stepwise regression. Backward elimination starts from the full model $y = X\beta + \varepsilon$ and identifies a variable x_k such that

1. its omission results in smallest increase of RSS; or
2. it has the smallest t -statistics $t_k = b_k^{\text{LS}} / \sqrt{s_k^2 / (n - p)}$, where s_k^2 is an estimate of b_k^{LS} variance, or any other test statistics of $H_0 : \beta_{0k} = 0$; or
3. its removal causes smallest change of prediction or information criteria characterizing fit or prediction power of the model. A well-known examples of information criteria are modified coefficient of determination $\bar{R}^2 = 1 - (n + p)e^{\top}e/n(n - p)$, Akaike information criterion (Akaike, 1974), $\text{AIC} = \log(e^{\top}e/n) + 2p/n$, and Schwarz information criterion (Schwarz, 1978), $\text{SIC} = \log(e^{\top}e/n) + p \ln n/n$, where n and p represents sample size and the number of regressors, respectively.

Next, the variable x_k is excluded from regression by setting $b_k = 0$ if (1) one did not reach a pre-specified number of variables yet or (2) the test statistics or change of the information criterion lies below some selected significance level.

Before discussing properties of backward elimination, let us make several notes on information criteria used for the elimination and their optimality. There is a wide range of selection criteria, including classical AIC, SIC, FPE_λ by Shibata (1984), cross validation by Stone (1974), and so on. Despite one can consider the same measure of the optimality of variable selection, such as the sum of squared prediction errors Shibata (1981), one can often see contradictory results concerning the selection criteria (see Li (1987) and Shao (1993); or Shibata (1981) and Rao and Wu (1989)). This is caused by different underlying assumptions about the true model Shao (1997). Some criteria, such as AIC and cross validation, are optimal if one assumes that there is no finite-dimensional true model (i.e., the number of variables increases with the sample size); see Shibata (1981) and Li (1987). On the other hand, some criteria, such as SIC, are consistent if one assumes that there is a true model with a finite number of variables; see Rao and Wu (1989) and Shao (1997). Finally, note that even though some criteria, being optimal in the same sense, are asymptotically equivalent, their finite sample properties can differ substantially. See Chap. III.1 for more details.

Let us now return back to backward elimination, which can be also viewed as a pre-test estimator Judge and Bock (1983). Although it is often used in practice, it involves largely arbitrary choice of the significance level. In addition, it has rather poor statistical properties caused primarily by discontinuity of the selection decision, see Magnus (1999). Moreover, even if a stepwise procedure is employed, one should take care of reporting correct variances and confidence intervals valid for the whole decision sequence. Inference for the finally selected model as if it were the only model considered leads to significant biases, see Danilov and Magnus (2004), Weiss (1995) and Zhang (1992). Backward elimination also does not perform well in the presence of multicollinearity and it cannot be used if $p > n$. Finally, let us note that a nearly optimal and admissible alternative is proposed in Magnus (2002).

Forward Selection

Backward elimination cannot be applied if there are more variables than observations, and additionally, it may be very computationally expensive if there are many variables. A classical alternative is *forward selection*, where one starts from an intercept-only model and adds one after another variables that provide the largest decrease of RSS. Adding stops when the F -statistics

$$R = \frac{\text{RSS}_p - \text{RSS}_{p+1}}{\text{RSS}_{p+1}}(n - p - 2)$$

lies below a pre-specified critical ‘F-to-enter’ value. The forward selection can be combined with the backward selection (e.g., after adding a variable, one performs one step of backward elimination), which is known as a stepwise regression Efromyson (1960). Its computational complexity is discussed in Miller (2002).

Note that most disadvantages of backward elimination apply to forward selection as well. In particular, correct variances and confidence intervals should be reported, see Miller (2002) for their approximations. Moreover, forward selection can be overly aggressive in selection in the respect that if a variable x is already included in a model, forward selection primarily adds variables orthogonal to x , thus ignoring possibly useful variables that are correlated with x . To improve upon this, Efron et al. (2004) proposed least angle regression, considering correlations of to-be-added variables jointly with respect to all variables already included in the model.

All-Subsets Regression

Neither forward selection, nor backward elimination guarantee the optimality of the selected submodel, even when both methods lead to the same results. This can happen especially when a pair of variables has jointly high predictive power; for example, if the dependent variable y depends on the difference of two variables $x_1 - x_2$. An alternative approach, which is aiming at optimality of the selected subset of variables – *all-subsets regression* – is based on forming a model for each subset of explanatory variables. Each model is estimated and a selected prediction or information criterion, which quantifies the unexplained variation of the dependent variable and the parsimony of the model, is evaluated. Finally, the model attaining the best value of a criterion is selected and variables missing in this model are omitted.

This approach deserves several comments. First, one can use many other criteria instead of AIC or SIC. These could be based on the test statistics of a joint hypothesis that a group of variables has zero coefficients, extensions or modifications of AIC or SIC, general Bayesian predictive criteria, criteria using non-sample information, model selection based on estimated parameter values at each subsample and so on. See the next subsection, Bedrick and Tsai (1994), Hughes and Maxwell (2003), Jiang and Liu (2004), Ibrahim and Ming-Hui (1997), Shao (1997), Shi and Tsai (1998), Zheng and Loh (1995), for instance, and Chap. III.1 for a more detailed overview.

Second, the evaluation and estimation of all submodels of a given regression model can be very computationally intensive, especially if the number of variables is large. This motivated tree-like algorithms searching through all submodels, but once they reject a submodel, they automatically reject all models containing only a subset of variables of the rejected submodel, see Edwards and Havranek (1987). These so-called branch-and-bound techniques are discussed in Miller (2002), for instance.

An alternative computational approach, which is increasingly used in applications where the number of explanatory variables is very large, is based on the *genetic programming* (genetic algorithm, GA) approach, see Wasserman and Sudjianto (1994). Similarly to branch-and-bound methods, GAs perform a non-exhaustive search through the space of all submodels. The procedure works as follows. First, each submodel which is represented by a “chromosome” – a $p \times 1$ vector $\mathbf{m}_j = \{I_1^j, \dots, I_p^j\} \in \{0, 1\}^p$ of indicators, where I_k^j indicates whether the k th variable is included in the submodel defined by \mathbf{m}_j . Next, to find the best submodel, one starts with an (initially randomly selected) population $\mathcal{P} = \{\mathbf{m}_j\}_{j=1}^J$ of submodels that are compared with each other in terms of information or prediction criteria. Further, this population \mathcal{P} is iteratively modified: in each step, pairs of submodels $\mathbf{m}_j, \mathbf{m}_{j'} \in \mathcal{P}$ combine their characteristics (chromosomes) to create their offsprings \mathbf{m}_j^* . This process can have many different forms such as $\mathbf{m}_j^* = (\mathbf{m}_j + \mathbf{m}_{j'} + \mathbf{r}_m) \bmod 1$ or $\mathbf{m}_j^* = (1, \dots, 1, 0, \dots, 0)^\top \mathbf{m}_j + (0, \dots, 0, 1, \dots, 1)^\top \mathbf{m}_{j'} + \mathbf{r}_m$, where \mathbf{r}_m is a possibly non-zero random mutation. Whenever an offspring \mathbf{m}_j^* performs better than its “parent” models $\mathbf{m}_j, \mathbf{m}_{j'}$, \mathbf{m}_j^* replaces \mathbf{m}_j in population \mathcal{P} . Performing this action for all $j = 1, \dots, J$ creates a new population. By repeating this population-renewal, GAs search through the space of all available submodels and keep only the best ones in the population \mathcal{P} . Thus, GAs provide a rather effective way of obtaining the best submodel, especially when the number of explanatory variables is very high, since the search is not exhaustive. See Chap. II.6 and Chambers (1998) for a more detailed introduction to genetic programming.

Cross Validation

Cross validation (CV) is a general model-selection principle, proposed already in Stone (1974), which chooses a specific model in a similar way as the prediction criteria. CV compares models, which can include all variables or exclude some, based on their out-of-sample performance, which is measured typically by MSE. To achieve this, a sample is split to two disjunct parts: one part is used for estimation and the other part serves for checking the fit of the estimated model on “new” data (i.e., data which were not used for estimation) by comparing the observed and predicted values.

Probably the most popular variant is the leave-one-out cross-validation (LOU CV), which can be used not only for model selection, but also for choosing nuisance parameters (e.g., in nonparametric regression; see Härdle (1992)). Assume we have a set of models $\mathbf{y} = h_k(\mathbf{X}, \boldsymbol{\beta}) + \boldsymbol{\varepsilon}$ defined by regression functions $h_k, k = 1, \dots, M$,

that determine variables included or excluded from regression. For model given by h_k , LOU CV evaluates

$$CV_k = \sum_{i=1}^n (y_i - \hat{y}_{i,-i})^2, \quad (8.7)$$

where $\hat{y}_{i,-i}$ is the prediction at x_i based on the model $y_{-i} = h_k(\mathbf{X}_{-i}, \boldsymbol{\beta}) + \boldsymbol{\varepsilon}_{-i}$ and $y_{-i}, \mathbf{X}_{-i}, \boldsymbol{\varepsilon}_{-i}$ are the vectors and matrices $y, \mathbf{X}, \boldsymbol{\varepsilon}$ without their i th elements and rows, respectively. Thus, all but the i th observation are used for estimation and the i th observation is used to check the out-of-sample prediction. Having evaluated CV_k for each model, $k = 1, \dots, M$, we select the model commanding the minimum $\min_{k=1, \dots, M} CV_k$.

Unfortunately, LOU CV is not consistent as far as the linear model selection is concerned. To make CV a consistent model selection method, it is necessary to omit n_v observations from the sample used for estimation, where $\lim_{n \rightarrow \infty} n_v/n = 1$. This fundamental result derived in Shao (1993) places a heavy computational burden on the CV model selection. Since our main use of CV in this chapter concerns nuisance parameter selection, we do not discuss this type of CV any further. See Miller (2002) and Chap. III.1 for further details.

Example 1 We compare several mentioned variable selection methods using a classical data set on air pollution used originally by McDonald and Schwing (1973), who modeled mortality depending on 15 explanatory variables ranging from climate and air pollution to socioeconomic characteristics and who additionally demonstrated instabilities of LS estimates using this data set. We refer to the explanatory variables of data Pollution simply by numbers 1 to 15.

We applied the forward, backward, and all-subset selection procedures to this data set. The results reported in Table 8.1 demonstrate that although all three methods could lead to the same subset of variables (e.g., if we search a model consisting

1

Table 8.1. Variables selected from Pollution data by different selection procedures. RSS is in brackets

Number of variables	Forward selection	Backward elimination	All-subset selection
1	9 (133,695)	9 (133,695)	9 (133,695)
2	6, 9 (99,841)	6, 9 (99,841)	6, 9 (99,841)
3	2, 6, 9 (82,389)	2, 6, 9 (82,389)	2, 6, 9 (82,389)
4	2, 6, 9, 14 (72,250)	2, 5, 6, 9 (74,666)	1, 2, 9, 14 (69,154)
5	1, 2, 6, 9, 14 (64,634)	2, 6, 9, 12, 13 (69,135)	1, 2, 6, 9, 14 (64,634)

of two or three variables), this is not the case in general. For example, searching for a subset of four variables, the variables selected by backward and forward selection differ, and in both cases, the selected model is suboptimal (compared to all-subsets regression) in the sense of the unexplained variance measured by RSS.

8.1.4 Principle Components Regression

In some situations, it is not feasible to use variable selection to reduce the number of explanatory variables or it is not desirable to do so. The first case can occur if the number of explanatory variables is large compared to the number of observations. The latter case is typical in situations when we observe many characteristics of the same type, for example, temperature or electro-impulse measurements from different sensors on a human body. They could be possibly correlated with each other and there is no a priori reason why measurements at some points of a skull, for instance, should be significant while other ones would not be important at all. Since such data typically exhibit (exact) multicollinearity and we do not want to exclude some or even majority of variables, we have to reduce the dimension of the data in another way.

A general method that can be used both under near and exact multicollinearity is based on the *principle components analysis* (PCA), see Chap. III.6. Its aim is to reduce the dimension of explanatory variables by finding a small number of linear combinations of explanatory variables X that capture most of the variation in X and to use these linear combinations as new explanatory variables instead the original one. Suppose that G is an orthonormal matrix that diagonalizes matrix $X^T X$: $G^T G = I$, $X^T X = G \Lambda G^T$, and $G^T X^T X G = \Lambda$, where $\Lambda = \text{diag}(\lambda_1, \dots, \lambda_p)$ is a diagonal matrix of eigenvalues of $X^T X$.

3 **Definition 3** Assume without loss of generality that $\lambda_1 \geq \dots \geq \lambda_p$ and g_1, \dots, g_p are the corresponding eigenvectors (columns of matrix G). Vector $z_i = X g_i$ for $i = 1, \dots, p$ such that $\lambda_i > 0$ is called the i th principle component (PC) of X and g_i represents the corresponding loadings.

PCA tries to approximate the original matrix X by projecting it into the lower-dimensional space spanned by the first k eigenvectors g_1, \dots, g_k . It can be shown that these projections capture most of the variability in X among all linear combinations of columns of X , see Härdle and Simar (2003).

2 **Theorem 2** There is no standardized linear combination $X a$, where $\|a\| = 1$, that has strictly larger variance than $z_1 = X g_1$: $\text{Var}(X a) \leq \text{Var}(z_1) = \lambda_1$. Additionally, the variance of the linear combination $z = X a$, $\|a\| = 1$, that is uncorrelated with the first k principle components z_1, \dots, z_k is maximized by the $(k + 1)$ -st principle component $z = z_{k+1}$ and $a = g_{k+1}$, $k = 1, \dots, p - 1$.

Consequently, one chooses a number k of PCs that capture a sufficient amount of data variability. This can be done by looking at the ratio $L_k = \sum_{i=1}^k \lambda_i / \sum_{i=1}^p \lambda_i$, which quantifies the fraction of the variance captured by the first k PCs compared to the total variance of X .

In the regression context, the chosen PCs are used as new explanatory variables, and consequently, PCs with small eigenvalues can be important too. Therefore, one can alternatively choose the PCs that exhibit highest correlation with the dependent variable y because the aim is to use the selected PCs for regressing the dependent variable y on them, see Jolliffe (1982). Moreover, for selecting “explanatory” PCs, it is also possible to use any variable selection method discussed in Sect. 8.1.3. Recently, Hwang and Nettleton (2003) proposed a new data-driven PC selection for PCR obtained by minimizing MSE.

Next, let us assume we selected a small number k of PCs $Z_k = (z_1, \dots, z_k)^\top$ by some rule such that matrix $Z_k^\top Z_k$ has a full rank, $k \leq p$. Then the *principle components regression* (PCR) is performed by regressing the dependent variable y on the selected PCs Z_k , which have a (much) smaller dimension than original data X , and consequently, multicollinearity is diminished or eliminated, see Gunst and Mason (1980). We estimate this new model by LS,

$$y = Z_k \boldsymbol{\gamma} + \eta = X G_k \boldsymbol{\gamma} + \eta,$$

where $G_k = (g_1, \dots, g_k)^\top$. Comparing it with the original model (8.1) shows that $\boldsymbol{\beta} = G_k \boldsymbol{\gamma}$. It is important to realize that in PCR we first fix G_k by means of PCA and then estimate $\boldsymbol{\gamma}$.

Finally, concerning different PC selection criteria, Barros and Rutledge (1998) demonstrates the superiority of the correlation-based PCR (CPCR) and convergence of many model-selection procedures toward the CPCR results. See also Depczynski et al. (2000) for a similar comparison of CPCR and PCR based on GA variable selection.

Example 2 Let us use data Pollution to demonstrate several important issues concerning PCR. First, we identify PCs of the data. The fraction of variance explained by the first k PCs as a function of k is depicted on Fig. 8.1 (dashed line). On the one side, almost all of the X variance is captured by the first PC. On the other side, the percentage of the y variance explained by the first k PCs (solid line) grows and reaches its maximum relatively slowly. Thus, the inclusion of about 7 PCs seems to be necessary when using this strategy.

On the other hand, using some variable selection method or checking the correlation of PCs with the dependent variable y reveals that PCs 1, 3, 4, 5, 7 exhibit highest correlations with y (higher than 0.25), and naturally, a model using these 5 PCs has more explanatory power ($\bar{R}^2 = 0.70$) than for example the first 6 PCs together ($\bar{R}^2 = 0.65$). Thus, considering not only PCs that capture most of the X variability, but also those having large correlations with the dependent variable enables building more parsimonious models.

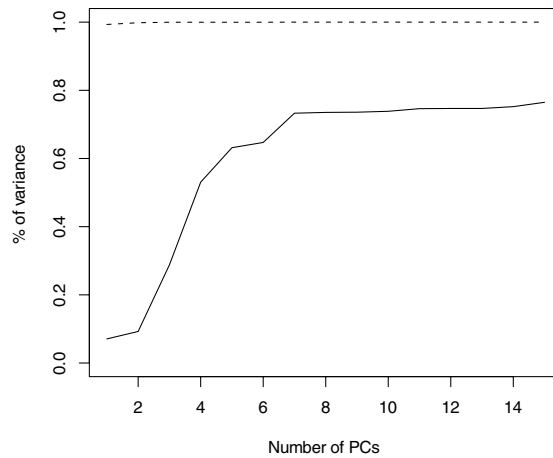


Figure 8.1. Fraction of the explained variance of X (dashed line) and y (solid line) by the first k PCs

8.1.5 Shrinkage Estimators

We argued in Sect. 8.1.2 that an alternative way of dealing with unpleasant consequences of multicollinearity lies in biased estimation: we can sacrifice a small bias for a significant reduction in variance of an estimator so that its MSE decreases. Since it holds for an estimator b and a real constant $c \in \mathbb{R}$ that $\text{Var}(c\hat{\beta}) = c^2 \text{Var}(\hat{\beta})$, a bias of the estimator $\hat{\beta}$ towards zero, $|c| < 1$, naturally leads to a reduction in variance. This observation motivates a whole class of biased estimators – *shrinkage estimators* – that are biased towards zero in all or just some of their components. In other words, they “shrink” the Euclidean norm of estimates compared to that of the corresponding unbiased estimate. This is perhaps easiest to observe on the example of the Stein-rule estimator, which can be expressed in linear regression model (8.1) as

$$\hat{\beta}^{\text{SR}} = \left(1 - \frac{k e^{\top} e}{n \hat{\beta}^{\text{LS}\top} X^{\top} X \hat{\beta}^{\text{LS}}} \right) \hat{\beta}^{\text{LS}}, \quad (8.8)$$

where $k > 0$ is an arbitrary scalar constant and $e^{\top} e/n$ represents an estimate of the residual variance Gruber (1998). Apparently, the Stein-rule estimator just multiplies the LS estimator by a constant smaller than one. See Gruber (1998) and Judge and Bock (1983) for an overview of this and many other biased estimators.

In the following subsections, we discuss various shrinkage estimators that perform well under multicollinearity and that can possibly act as variable selection tools as well: the ridge regression estimator and its modifications (Sect. 8.1.6), continuum regression (Sect. 8.1.7), the Lasso estimator and its variants (Sect. 8.1.8), and partial least squares (Sect. 8.1.9). Let us note that there are also other shrinkage estimators, which either do not perform well under various forms of multicollinearity

(e.g., Stein-rule estimator) or are discussed in other parts of this chapter (e.g., pre-test and PCR estimators in Sects. 8.1.3 and 8.1.4, respectively).

Ridge Regression

8.1.6

Probably the best known shrinkage estimator is the *ridge estimator* proposed and studied by Hoerl and Kennard (1970). Having a non-orthogonal or even nearly singular matrix $X^T X$, one can add a positive constant $k > 0$ to its diagonal to improve conditioning.

Definition 4 Ridge regression (RR) estimator is defined for model (8.1) by

4

$$\hat{\beta}^{RR} = (X^T X + kI)^{-1} X^T y \quad (8.9)$$

for some ridge parameter $k > 0$.

“Increasing” the diagonal of $X^T X$ before inversion shrinks $\hat{\beta}^{RR}$ compared to $\hat{\beta}^{LS}$ and introduces a bias. Additionally, Hoerl and Kennard (1970) also showed that the derivative of $MSE(\hat{\beta}^{RR})$ with respect to k is negative at $k = 0$. This indicates that the bias

$$\text{Bias}(\hat{\beta}^{RR}) = -k (X^T X + kI)^{-1} \beta$$

can be smaller than the decrease in variance (here for a homoscedastic linear model with error variance σ^2)

$$\text{Var}(\hat{\beta}^{RR}) - \text{Var}(\hat{\beta}^{LS}) = \sigma^2 (X^T X + kI)^{-1} X^T X (X^T X + kI)^{-1} - \sigma^2 (X^T X)^{-1}$$

caused by shrinking at least for some values of k . The intervals for k where RR dominates LS are derived, for example, in Chawla (1990), Gruber (1998) and Rao and Toutenberg (1999). Moreover, the improvement in $MSE(\hat{\beta}^{RR})$ with respect to $MSE(\hat{\beta}^{LS})$ is significant under multicollinearity while being negligible for nearly orthogonal systems. A classical result for model (8.1) under $\epsilon \sim N(0, \sigma^2 I_n)$ states that $MSE(\hat{\beta}^{RR}) - MSE(\hat{\beta}^{LS}) < 0$ is negative definite if $k < k_{\max} = 2\sigma^2 / \beta^T \beta$, see Vinod and Ullah (1981), where an operational estimate of k_{\max} is discussed too. Notice however that the conditions for the dominance of the RR and other some other shrinkage estimators over LS can look quite differently in the case of non-normal errors Ullah et al. (1983).

In applications, an important question remains: how to choose the ridge parameter k ? In the original paper by Hoerl and Kennard (1970), the use of the ridge trace, a plot of the components of the estimated $\hat{\beta}^{RR}$ against k , was advocated. If data exhibit multicollinearity, one usually observes a region of instability for k close to zero and then stable estimates for large values of ridge parameter k . One should choose the smallest k lying in the region of stable estimates. Alternatively,

one could search for k minimizing $\text{MSE}(\hat{\boldsymbol{\beta}}^{RR})$; see the subsection on generalized RR for more details. Furthermore, many other methods for model selection could be employed too; for example, LOU CV (Sect. 8.1.3) performed on a grid of k values is often used in this context.

Statistics important for inference based on RR estimates are discussed in Hoerl and Kennard (1970) and Vinod and Ullah (1981) both for the case of a fixed k as well as in the case of some data-driven choices. Moreover, the latter work also describes algorithms for a fast and efficient RR computation.

To conclude, let us note that the RR estimator $\hat{\boldsymbol{\beta}}^{RR}$ in model (8.1) can be also defined as a solution of a restricted minimization problem

$$\hat{\boldsymbol{\beta}}^{RR} = \underset{\hat{\boldsymbol{\beta}}: \|\hat{\boldsymbol{\beta}}\|_2^2 \leq r^2}{\text{argmin}} (\mathbf{y} - \mathbf{X}\hat{\boldsymbol{\beta}})^\top (\mathbf{y} - \mathbf{X}\hat{\boldsymbol{\beta}}), \quad (8.10)$$

or equivalently as

$$\hat{\boldsymbol{\beta}}^{RR} = \underset{\hat{\boldsymbol{\beta}}}{\text{argmin}} (\mathbf{y} - \mathbf{X}\hat{\boldsymbol{\beta}})^\top (\mathbf{y} - \mathbf{X}\hat{\boldsymbol{\beta}}) + k\|\hat{\boldsymbol{\beta}}\|_2^2, \quad (8.11)$$

where r represents a tuning parameter corresponding to k (Swamy et al., 1978). This formulation was used by Ngo et al. (2003), for instance. Moreover, (8.10) reveals one controversial issue in RR: rescaling of the original data to make $\mathbf{X}^\top \mathbf{X}$ a correlation matrix. Although there are no requirements of this kind necessary for theoretical results, standardization is often recommended to make influence of the constraint $\|\hat{\boldsymbol{\beta}}\|_2^2 \leq r^2$ same for all variables. There are also studies showing adverse effects of this standardization on estimation, see Vinod and Ullah (1981) for a discussion. A possible solution is generalized RR, which assigns to each variable its own ridge parameter (see the next paragraph).

Generalized Ridge Regression

The RR estimator can be generalized in the sense that each diagonal element of $\mathbf{X}^\top \mathbf{X}$ is modified separately. To achieve that let us recall that this matrix can be diagonalized: $\mathbf{X}^\top \mathbf{X} = \mathbf{G}^\top \boldsymbol{\Lambda} \mathbf{G}$, where \mathbf{G} is an orthonormal matrix and $\boldsymbol{\Lambda}$ is a diagonal matrix containing eigenvalues $\lambda_1, \dots, \lambda_p$.

5 **Definition 5** Generalized ridge regression (GRR) estimator is defined for model (8.1) by

$$\hat{\boldsymbol{\beta}}^{GRR} = (\mathbf{X}^\top \mathbf{X} + \mathbf{G} \mathbf{K} \mathbf{G}^\top)^{-1} \mathbf{X}^\top \mathbf{y} \quad (8.12)$$

for a diagonal matrix $\mathbf{K} = \text{diag}(k_1, \dots, k_p)$ of ridge parameters.

The main advantage of this generalization being ridge coefficients specific to each variable, it is important to know how to choose the matrix \mathbf{K} . In Hoerl and Kennard (1970) the following result is derived.

Theorem 3 Assume that X in model (8.1) has a full rank, $\boldsymbol{\varepsilon} \sim N(0, \sigma^2 \mathbf{I}_n)$, and $n > p$. Further, let $X = \mathbf{H}\boldsymbol{\Lambda}^{1/2}\mathbf{G}^\top$ be the singular value decomposition of X and $\boldsymbol{\gamma} = \mathbf{G}^\top \boldsymbol{\beta}_0$. The MSE-minimizing choice of K in (8.12) is $K = \sigma^2 \text{diag}(\gamma_1^{-2}, \dots, \gamma_p^{-2})$.

An operational version (feasible GRR) is based on an unbiased estimate $\widehat{\boldsymbol{\gamma}}_i = \mathbf{G}^\top \widehat{\boldsymbol{\beta}}^{\text{LS}}$ and $s^2 = (\mathbf{y} - \mathbf{H}\widehat{\boldsymbol{\gamma}})^\top (\mathbf{y} - \mathbf{H}\widehat{\boldsymbol{\gamma}})$. See Hoerl and Kennard (1970) and Vinod and Ullah (1981), where you also find the bias and MSE of this operational GRR estimator, and Wang and Chow (1990) for further extensions of this approach. Let us note that the feasible GRR (FGRR) estimator does not have to possess the MSE-optimality property of GRR because the optimal choice of K is replaced by an estimate. Nevertheless, the optimality property of FGRR is preserved if $\lambda_i \gamma_i^2 \leq 2\sigma^2$, where λ_i is the (i, i) -th element of $\boldsymbol{\Lambda}$ (Farebrother, 1976).

Additionally, given an estimate of MSE-minimizing $\widehat{K} = \text{diag}(\widehat{k}_1, \dots, \widehat{k}_p)$, many authors proposed to choose the ridge parameter k in ordinary RR as a harmonic mean of \widehat{k}_i , $i = 1, \dots, p$; see Hoerl et al. (1975), for instance.

Almost Unbiased Ridge Regression

Motivated by results on GRR, Kadiyala (1984) proposed to correct GRR for its bias using the first-order bias approximation. This yields almost unbiased GRR (AUGRR) estimator

$$\widehat{\boldsymbol{\beta}}^{\text{AUGRR}} = (\mathbf{X}^\top \mathbf{X} + \mathbf{G}\widehat{K}\mathbf{G}^\top)^{-1} (\mathbf{X}^\top \mathbf{y} + \mathbf{K}\mathbf{G}^\top \boldsymbol{\beta}_0) .$$

The true parameter value $\boldsymbol{\beta}_0$ being unknown, Ohtani (1986) defined a feasible AUGRR estimator by replacing the unknown $\boldsymbol{\beta}_0$ by $\widehat{\boldsymbol{\beta}}^{\text{FGRR}}$ and K by the employed ridge matrix. Additionally, a comparison of the FGRR and feasible AUGRR estimators with respect to MSE proved that FGRR has a smaller MSE than AUGRR in a wide range of parameter space. Similar observation was also done under a more general loss function in Wan (2002). Furthermore, Akdeniz et al. (2004) derived exact formulas for the moments of the feasible AUGRR estimator.

Further Extensions

RR can be applied also under exact multicollinearity, which arises for example in data with more variables than observations. Although the theory and application of RR is the same as in the case of full-rank data, the computational burden of $O(np^2 + p^3)$ operations becomes too high for $p > n$. A faster algorithm with computational complexity only $O(np^2)$ was found by Hawkins and Yin (2002).

Moreover, there are many further extensions of the RR principle that go beyond the extent of this chapter. To mention at least some of them, let us refer a reader to works comparing or combining various ridge and shrinkage approaches Kibria (1996); Shiaishi and Konno (1995); Singh et al. (1994) and to monograph by Gruber (1998).

- 3 **Example 3** Using data Pollution once again, we estimated RR for ridge parameter $k \in (0, 10)$ and plotted the estimated coefficients $\hat{\beta}^{RR}$ as functions of k (ridge trace plot), see Fig. 8.2. For the sake of simplicity, we restricted ourselves only to variables that were selected by some variable selection procedure in Table 8.1 (1, 2, 6, 9, 12, 13, 14). The plot shows the effect of ridge parameter k on slope estimates ($k = 0$ corresponds to LS). Apparently, slopes of some variables are affected very little (e.g., variable 1), some significantly (e.g., the magnitude of variable 14 increases more than twice), and some variables shrink extremely (e.g., variables 12 and 13). In all cases, the biggest change occurs between $k = 0$ and $k = 2$ and estimates gradually stabilize for $k > 2$. The vertical dashed line in Fig. 8.2 represents the CV estimate of k ($k_{CV} = 6.87$).

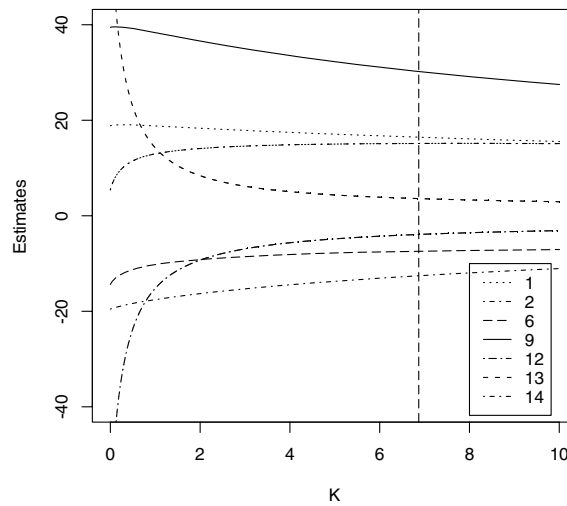


Figure 8.2. Ridge trace plot for variables 1, 2, 6, 9, 12, 13, 14 of data Pollution. The vertical line represents the CV-choice of k

8.1.7 Continuum Regression

RR discussed in Sect. 8.1.6 is very closely connected with the *continuum regression* proposed by Brooks and Stone (1990) as a unifying approach to the LS, PCR, and partial least squares (see Sect. 8.1.9) estimation.

- 6 **Definition 6** A continuum regression (CR) estimator $\hat{\beta}^{CR}(\alpha)$ of model (8.1) is a coefficient vector maximizing function

$$T_{\alpha}(\mathbf{c}) = (\mathbf{c}^{\top} \mathbf{s})^2 (\mathbf{c}^{\top} \mathbf{S} \mathbf{c})^{\alpha-1} = (\mathbf{c}^{\top} \mathbf{X}^{\top} \mathbf{y})^2 (\mathbf{c}^{\top} \mathbf{X}^{\top} \mathbf{X} \mathbf{c})^{\alpha-1}, \quad (8.13)$$

for a given value of parameter $\alpha \geq 0$ and a given length $\|c\|$, where $S = X^T X$ and $s = X^T y$.

This definition yields estimates proportional to LS for $\alpha = 0$, to PCR for $\alpha \rightarrow \infty$, and to yet-to-be-discussed partial least squares for $\alpha = 1$. Apart from this, the advantage of CR is that one can adaptively select among the methods by searching an optimal α . To determine α , Brooks and Stone (1990) used CV.

The relationship between RR and CR was indicated already in Sundberg (1993), but the most important result came after uncovering possible discontinuities of CR estimates as a function of data and α by Björkström and Sundberg (1996). In an attempt to remedy the discontinuity of the original CR, Björkström and Sundberg (1999) not only proposed to maximize

$$T_\delta(c) = (c^T s)^2 (c^T S c)^{-1} |c^T S c + \delta|^{-1} ,$$

for $\delta \geq 0$ instead of $T_\alpha(c)$ from Definition 6 (δ can be chosen by CV), but also proved the following proposition.

Theorem 4 If a regressor b_f is defined according to

4

$$b_f = \operatorname{argmax}_{\|c\|=1} f \{K^2(c), V(c)\} ,$$

where $K(c) = y^T X c$, $V(c) = \|X c\|^2$, $f(K^2, V)$ is increasing in K^2 for constant V , and increasing in V for constant K^2 , and finally, if $X^T y$ is not orthogonal to all eigenvectors corresponding to the largest eigenvalue λ_{\max} of $X^T X$, then there exists a number $k \in (-\infty, \lambda_{\max}) \cup [0, +\infty]$ such that b_f is proportional to $(X^T X + kI)^{-1} X^T y$, including the limiting cases $k \rightarrow 0$, $k \rightarrow \pm\infty$, and $k \rightarrow -\lambda_{\max}$.

Thus, the RR estimator fundamentally underlies many methods dealing with multicollinear and reduced rank data such as mentioned PCR and partial least squares. Notice however that negative values of the ridge coefficient k have to be admitted here.

Finally, let us note that CR can be extended to multiple-response-variables models (Brooks and Stone, 1994).

Lasso

8.1.8

The ridge regression discussed in Sect. 8.1.6 motivates another shrinkage method: *Lasso* (least absolute shrinkage and selection operator) by Tibshirani (1996). Formulation (8.10) states that RR can be viewed as a minimization with respect to an upper bound on the L_2 norm of estimate $\|\hat{\beta}\|_2$. A natural extension is to consider constraints on the L_q norm $\|\hat{\beta}\|_q$, $q > 0$. Specifically, Tibshirani (1996) studied case of $q = 1$, that is L_1 norm.

7

Definition 7 The Lasso estimator for the regression model (8.1) is defined by

$$\hat{\beta}^L = \underset{\|\beta\|_1 \leq r}{\operatorname{argmin}} (y - X\beta)^\top (y - X\beta) , \quad (8.14)$$

where $r \geq 0$ is a tuning parameter.

Lasso is a shrinkage estimator that has one specific feature compared to ordinary RR. Because of the geometry of L_1 -norm restriction, Lasso shrinks the effect of some variables and eliminates influence of the others, that is, sets their coefficients to zero. Thus, it combines regression shrinkage with variable selection, and as Tibshirani (1996) demonstrated also by means of simulation, it compares favorably to all-subsets regression. In this context, it is interesting that Lasso could be formulated as a special case of least angle regression by Efron et al. (2004). Finally, let us note that to achieve the same kind of shrinking and variable-selection effects for all variables, they should be standardized before used in Lasso; see Miller (2002) for details.

As far as the inference for the Lasso estimator is concerned, Knight and Fu (2000) recently studied the asymptotic distribution of Lasso-type estimators using L_q -norm condition $\|\beta\|_q \leq r$ with $q \leq 1$, including behavior under nearly-singular designs.

Now, it remains to find out how Lasso estimates can be computed. Equation (8.14) indicates that one has to solve a restricted quadratic optimization problem. Setting $\beta_j^+ = \max\{\beta_j, 0\}$ and $\beta_j^- = -\min\{\beta_j, 0\}$, the restriction $\|\beta\| \leq r$ can be written as $2p + 1$ constraints: $\beta_j^+ \geq 0, \beta_j^- \geq 0$, and $\sum_{j=1}^p (\beta_j^+ - \beta_j^-) \leq r$. Thus, convergence is assured in $2p + 1$ steps. Additionally, the unknown tuning parameter r is to be selected by means of CV. Further, although solving (8.14) is straightforward in usual regression problems, it can become very demanding for reduced-rank data, $p > n$. Osborne et al. (1999) treated lasso as a convex programming problem, and by formulating its dual problem, developed an efficient algorithm usable even for $p > n$.

4

Example 4 Let us use data Pollution once more to exemplify the use of Lasso. To summarize the Lasso results, we use the same plot as Tibshirani (1996) and Efron et al. (2004) used, see Fig. 8.3. It contains standardized slope estimates as a function of the constraint $\|b\| \leq r$, which is represented by an index $r / \max \|\hat{\beta}\| = \|\hat{\beta}^L\| / \|\hat{\beta}^{LS}\|$ (the LS estimate $\hat{\beta}^{LS}$ corresponds to $\hat{\beta}^L$ under $r = \infty$, and thus, renders the maximum of $\|\hat{\beta}^L\|$). Moreover, to keep the graph simple, we plotted again only variables that were selected by variable selection procedures in Table 8.1 (1, 2, 6, 9, 12, 13, 14).

In Fig. 8.3, we can observe which variables are included in the regression (have a nonzero coefficient) as tuning parameter r increases. Clearly, the order in which the first of these variables become significant – 9, 6, 14, 1, 2 – closely resembles

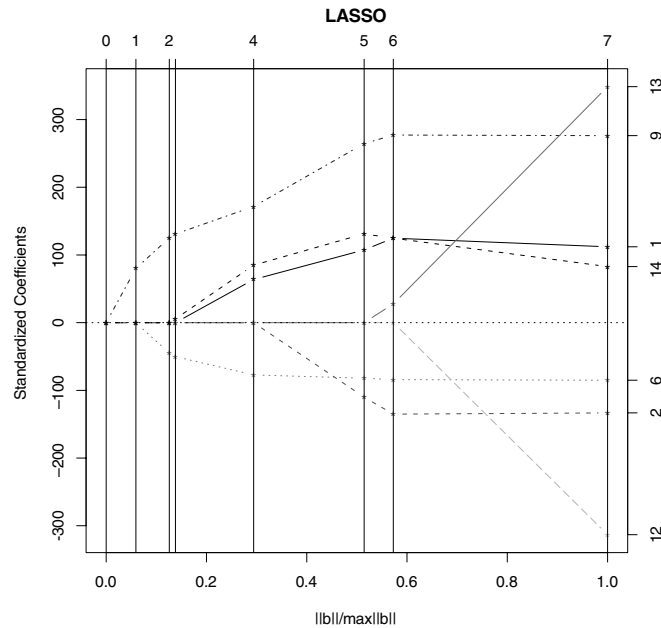


Figure 8.3. Slope coefficients for variables 1, 2, 6, 9, 12, 13, 14 of data Pollution estimated by Lasso at different constraint levels, $r/\max\|\hat{\beta}\|$. The *right axis* assigns to each line the number of variable it represents and the *top axis* indicates the number of variables included in the regression

the results of variable selection procedures in Table 8.1. Thus, Lasso combines shrinkage estimation and variable selection: at a given constraint level r , it shrinks coefficients of some variables and removes the others by setting their coefficients equal to zero.

Partial Least Squares

8.1.9

A general modeling approach to most of the methods covered so far was CR in Sect. 8.1.7, whereby it has two “extremes”: LS for $\alpha = 0$ and PCR for $\alpha \rightarrow \infty$. The *partial least squares* (PLS) regression lies in between – it is a special case of (8.13) for $\alpha = 1$, see Brooks and Stone (1990). Originally proposed by Wold (1966), it was presented as an algorithm that searches for linear combinations of explanatory variables best explaining the dependent variable. Similarly to PCR, PLS also aims especially at situations when the number of explanatory variables is large compared to the number of observations. Here we present the PLS idea and algorithm themselves as well as the latest results on variable selection and inference in PLS.

Having many explanatory variables X , the aim of the PLS method is to find a small number of linear combinations $T_1 = Xc_1, \dots, T_q = Xc_q$ of these variables,

thought about as latent variables, explaining observed responses

$$\hat{\mathbf{y}} = \hat{\beta}_0 + \sum_{j=1}^q \mathbf{T}_j \hat{\beta}_j \quad (8.15)$$

(see Garthwaite (1994), and Helland (2001)). Thus, similarly to PCR, PLS reduces the dimension of data, but the criterion for searching linear combinations is different. Most importantly, it does not depend only on \mathbf{X} values, but on \mathbf{y} too.

Let us now present the PLS algorithm itself, which defines yet another shrinkage estimator as shown by Coutis (1996) and Jong (1995). (See Rao and Toutenberg, 1999 for more details and Garthwaite, 1994 for an alternative formulation.) The indices $\mathbf{T}_1, \dots, \mathbf{T}_q$ are constructed one after another. Estimating the intercept by $b_0 = \bar{y}$, let us start with centered variables $\mathbf{z}_0 = \mathbf{y} - \bar{y}$ and $\mathbf{U}_0 = \mathbf{X} - \bar{\mathbf{X}}$ and set $k = 1$.

1. Define the index $\mathbf{T}_k = \mathbf{U}_{k-1} (\mathbf{U}_{k-1}^\top \mathbf{z}_{k-1})^{-1} \mathbf{U}_{k-1}^\top \mathbf{z}_{k-1}$. This linear combination is given by the covariance of the unexplained part of the response variable \mathbf{z}_{k-1} and the unused part of explanatory variables \mathbf{U}_{k-1} .

2. Regress the current explanatory matrix \mathbf{U}_{k-1} on index \mathbf{T}_k

$$\mathbf{w}_k = (\mathbf{T}_k^\top \mathbf{T}_k)^{-1} \mathbf{T}_k^\top \mathbf{U}_{k-1}$$

and the yet-unexplained part of response \mathbf{z}_{k-1} on index \mathbf{T}_k

$$\hat{\beta}_k = (\mathbf{T}_k^\top \mathbf{T}_k)^{-1} \mathbf{T}_k^\top \mathbf{z}_{k-1},$$

thus obtaining the k th regression coefficient.

3. Compute residuals, that is the remaining parts of explanatory and response variables: $\mathbf{U}_k = \mathbf{U}_{k-1} - \mathbf{T}_k \mathbf{w}_k$ and $\mathbf{z}_k = \mathbf{z}_{k-1} - \mathbf{T}_k \hat{\beta}_k$. This implies that the indices \mathbf{T}_k and \mathbf{T}_l are not correlated for $k < l$.

4. Iterate by setting $k = k + 1$ or stop if $k = q$ is large enough.

This algorithm provides us with indices \mathbf{T}_k , which define the analogs of principle components in PCR, and the corresponding regression coefficients b_k in (8.15). The main open question is how to choose the number of components q . The original method proposed by Wold (1978) is based on cross validation. Provided that CV_k from (8.7) represents the CV index of PLS estimate with k factors, an additional index \mathbf{T}_{k+1} is added if Wold's R criterion $R = \text{CV}_{k+1}/\text{CV}_k$ is smaller than 1. This selects the first local minimum of the CV index, which is superior to finding the global minimum of CV_k as shown in Osten (1988). Alternatively, one can stop already when Wold's R exceeds 0.90 or 0.95 bound (modified Wold's R criteria) or to use other variable selection criteria such as AIC. In a recent simulation study, Li et al. (2002) showed that modified Wold's R is preferable to Wold's R and AIC. Furthermore, similarly to PCR, there are attempts to use GA for the component selection, see Leardi and González (1998) for instance.

Next, the first results on the asymptotic behavior of PLS appeared only during last decade. The asymptotic behavior of prediction errors was examined by Helland and Almoy (1994). The covariance matrix, confidence and prediction intervals based on PLS estimates were first studied by Denham (1997), but a more compact

expression was presented in Phatak et al. (2002). It is omitted here due to many technicalities required for its presentation. There are also attempts to find a sample-specific prediction error of PLS, which were compared by Faber et al. (2003).

Finally, note that there are many extensions of the presented algorithm, which is usually denoted PLS1. First of all, there are extensions (PLS2, SIMPLS, etc.) of PLS1 to models with multiple dependent variables, see Jong (1993) and Frank et al. (1993) for instance, which choose linear combinations (latent variables) not only within explanatory variables, but does the same also in the space spanned by dependent variables. A recent survey of these and other so-called two-block methods is given in Wegelin (2000). PLS was also adapted for on-line process modeling, see Qin (1997) for a recursive PLS algorithm. Additionally, in an attempt to simplify the interpretation of PLS results, Trygg and Wold (2002) proposed orthogonalized PLS. See Wold et al (2001) for further details on recent developments.

Example 5 Let us use again data Pollution, although it is not a typical application of PLS. As explained in Sects. 8.1.7 and 8.1.9, PLS and PCR are both based on the same principle (searching for linear combinations of original variables), but use different objective functions. To demonstrate, we estimated PLS for 1 to 15 latent variables and plotted the fraction of the X and y variance explained by the PLS latent variables in the same way as in Fig. 8.1. Both curves are in Fig. 8.4. Almost all of the variability in X is captured by the first latent variable, although this percentage is smaller than in the case of PCR. On the other hand, the percentage of the variance of y explained by the first k latent variables increases faster than in the case of PCR, see Fig. 8.4 (solid versus dotted line).

5

Comparison of the Methods

8.1.10

Methods discussed in Sects. 8.1.3–8.1.9 are aiming at the estimation of (nearly) singular problems and they are often very closely related, see Sect. 8.1.7. Here we provide several references to studies comparing the discussed methods.

First, an extensive simulation study comparing variable selection, PCR, RR, and PLS regression methods is presented in Frank et al. (1993). Although the results are conditional on the simulation design used in the study, they indicate that PCR, RR, and PLS are, in the case of ill-conditioned problems, highly preferable to variable selection. The differences between the best methods, RR and PLS, are rather small and the same holds for comparison of PLS and PCR, which seems to be slightly worse than RR. An empirical comparison of PCR and PLS was also done by Wentzell and Montoto (2003) with the same result. Next, the fact that neither PCR, nor PLS asymptotically dominates the other method was proved in Helland and Almoy (1994) and further discussed in Helland (2001). A similar asymptotic result was also given by Stoica and Söderström (1998). Finally, the fact that RR should not perform worse than PCR and PLS is supported by Theorem 4 in Sect. 8.1.7.

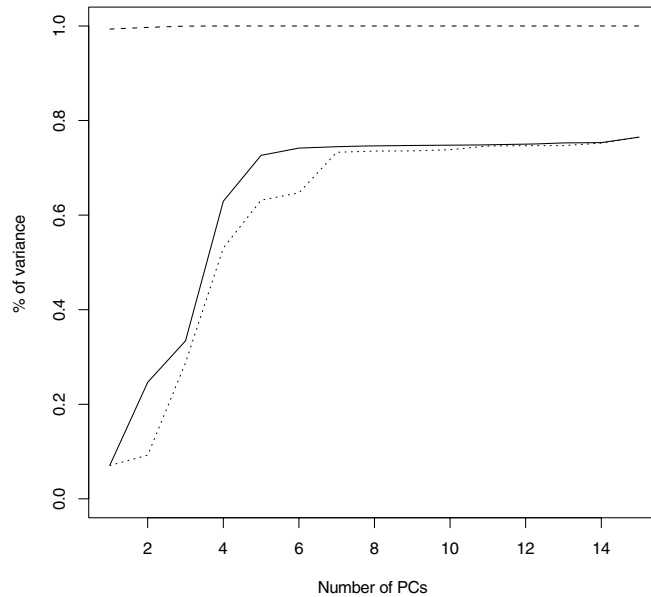


Figure 8.4. Fraction of the explained variance of X (dashed line) and y (solid line) by the first k latent variables in PLS regression and by first k PCs (dotted lines)

8.2

Nonlinear Regression Modeling

In this section, we study the nonlinear regression model

$$y_i = h(\mathbf{x}_i, \boldsymbol{\beta}_0) + \varepsilon_i, \quad (8.16)$$

$i = 1, \dots, n$, where $h : \mathbb{R}^p \times \mathbb{R}^k \rightarrow \mathbb{R}$ is a known regression function and $\boldsymbol{\beta}_0$ is a vector of k unknown parameters. Let us note that the methods discussed in this section are primarily meant for truly nonlinear models rather than intrinsically linear models. A regression model is called intrinsically linear if it can be unambiguously transformed to a model linear in parameters. For example, the regression model $y = \beta_1 x / (\beta_2 + x)$ can be expressed as $1/y = 1/\beta_1 + \beta_2/\beta_1 x$, which is linear in parameters $\theta_1 = 1/\beta_1$ and $\theta_2 = \beta_2/\beta_1$. Transforming a model to its linear form can often provide better inference, such as confidence regions, although one has to be aware of the effects of the transformation on the error-term distribution.

We first discuss the fitting and inference in the nonlinear regression (Sects. 8.2.1 and 8.2.2), whereby we again concentrate on the least square estimation. For an extensive discussion of theory and practice of nonlinear least squares regression see monographs Amemiya (1983), Bates and Watts (1988) and Seber and Wild (2003). Second, similarly to the linear modeling section, methods for ill-conditioned nonlinear systems are briefly reviewed in Sect. 8.2.3.

Fitting of Nonlinear Regression

8.2.1

In this section, we concentrate on estimating the vector β_0 of unknown parameters in (8.16) by *nonlinear least squares*.

Definition 8 The nonlinear least squares (NLS) estimator for the regression model (8.16) is defined by

8

$$\hat{\beta}^{\text{NLS}} = \underset{\beta \in \mathbb{R}^p}{\operatorname{argmin}} \sum_{i=1}^n \{y_i - \hat{y}_i(\beta)\}^2 = \underset{\beta \in \mathbb{R}^p}{\operatorname{argmin}} \sum_{i=1}^n \{y_i - h(x_i, \beta)\}^2. \quad (8.17)$$

Contrary to the linear model fitting, we cannot express analytically the solution of this optimization problem for a general function h . On the other hand, we can try to approximate the nonlinear objective function using the Taylor expansion because the existence of the first two derivatives of h is an often used condition for the asymptotic normality of NLS, and thus, could be readily assumed. Denoting $\mathbf{h}(\hat{\beta}) = \{h(x_i, \hat{\beta})\}_{i=1}^n$ and $S_n(\hat{\beta}) = \sum_{i=1}^n [y_i - h(x_i, \hat{\beta})]^2$, we can state the following theorem from Amemiya (1985).

Theorem 5 Let ε_i in (8.16) are independent and identically distributed with $E(\varepsilon|X) = 0$ and $\operatorname{Var}(\varepsilon|X) = \sigma^2 I_n$ and let B be an open neighborhood of β_0 . Further, assume that $h(x, \beta)$ is continuous on B uniformly with respect to x and twice continuously differentiable in B and that

5

1. $\lim_{n \rightarrow \infty} S_n(\beta) \neq 0$ for $\beta \neq \beta_0$;
2. $[\partial \mathbf{h}(\beta) / \partial \beta^\top]^\top [\partial \mathbf{h}(\beta) / \partial \beta^\top] / n$ converges uniformly in B to a finite matrix $A(\beta)$, such that $A(\beta_0)$ is nonsingular;
3. $\mathbf{h}(\beta^1)^\top [\partial^2 \mathbf{h}(\beta^2) / \partial \beta_j \partial \beta_k] / n$ converges uniformly for $\beta^1, \beta^2 \in B$ to a finite matrix for all $j, k = 1, \dots, k$.

Then the NLS estimator $\hat{\beta}^{\text{NLS}}$ is consistent and asymptotically normal

$$\sqrt{n} (\hat{\beta}^{\text{NLS}} - \beta_0) \rightarrow N(0, \sigma^2 A(\beta_0)^{-1}). \quad (8.18)$$

Hence, although there is no general explicit solution to (8.17), we can assume without loss of much generality that the objective function $S_n(\hat{\beta})$ is twice differentiable in order to devise a numerical optimization algorithm. The second-order Taylor expansion provides then a quadratic approximation of the minimized function, which can be used for obtaining an approximate minimum of the function, see Amemiya (1983). As a result, one should search in the direction of the steepest descent of a function, which is given by its gradient, to get a better approximation

of the minimum. We discuss here the incarnations of these methods specifically for the case of quadratic loss function in (8.17).

Newton's Method

The classical method based on the gradient approach is Newton's method, see Kennedy and Gentle (1980) and Amemiya (1983) for detailed discussion. Starting from an initial point $\hat{\beta}^1$, a better approximation is found by taking

$$\begin{aligned} \hat{\beta}^{k+1} &= \hat{\beta}^k - \mathbf{H}^{-1}(\mathbf{r}^2, \hat{\beta}^k) \mathbf{J}(\mathbf{r}, \hat{\beta}^k) = \\ &= \hat{\beta}^k - \left[\mathbf{J}(\mathbf{h}, \hat{\beta}^k)^\top \mathbf{J}(\mathbf{h}, \hat{\beta}^k) + \sum_{l=1}^n r_l(\hat{\beta}) \mathbf{H}(h_l, \hat{\beta}^k) \right]^{-1} \mathbf{J}(\mathbf{h}, \hat{\beta}^k)^\top \mathbf{r}(\hat{\beta}^k), \end{aligned} \quad (8.19)$$

where $\mathbf{r}(\beta) = \{[y_i - h(x_i, \beta)]\}_{i=1}^n$ represents the vector of residuals, $\mathbf{J}(\mathbf{f}, \beta) = \partial \mathbf{f}(\beta) / \partial \beta^\top$ is the Jacobian matrix of a vector function $\mathbf{f}(\beta)$, and $\mathbf{H}(\mathbf{f}, \beta) = \partial^2 \{\sum_{i=1}^n f_i(\beta)\} / \partial \beta \partial \beta^\top$ is the Hessian matrix of the sum of $\mathbf{f}(\beta)$.

To find $\hat{\beta}^{\text{NLS}}$, (8.19) is iterated until convergence is achieved. This is often verified by checking whether the relative change from $\hat{\beta}^k$ to $\hat{\beta}^{k+1}$ is sufficiently small. Unfortunately, this criterion can indicate a lack of progress rather than convergence. Instead, Bates and Watts (1988) proposed to check convergence by looking at some measure of orthogonality of residuals $\mathbf{r}(\hat{\beta}^k)$ towards the regression surface given by $\mathbf{h}(\hat{\beta}^k)$, since the identification assumption of model (8.16) is $E(\mathbf{r}(\beta_0) | \mathbf{X}) = 0$. See Björck (1996), Kennedy and Gentle (1980) and Thisted (1988) for more details and further modifications.

To evaluate iteration (8.19), it is necessary to invert the Hessian matrix $\mathbf{H}(\mathbf{r}^2, \beta)$. From the computational point of view, all issues discussed in Sect. 8.1 apply here too and one should use a numerically stable procedure, such as QR or SVD decompositions, to perform the inversion. Moreover, to guarantee that (8.19) leads to a better approximation of the minimum, that is $\mathbf{r}(\hat{\beta}^{k+1})^\top \mathbf{r}(\hat{\beta}^{k+1}) \leq \mathbf{r}(\hat{\beta}^k)^\top \mathbf{r}(\hat{\beta}^k)$, the Hessian matrix $\mathbf{H}(\mathbf{r}^2, \hat{\beta}^k)$ needs to be positive definite, which in general holds only in a neighborhood of β_0 (see the Levenberg–Marquardt method for a remedy). Even if it is so, the step in the gradient direction should not be too long, otherwise we “overshoot.” Modified Newton's method addresses this by using some fraction α_{k+1} of iteration step $\hat{\beta}^{k+1} = \hat{\beta}^k - \alpha_{k+1} \mathbf{H}^{-1}(\mathbf{r}^2, \hat{\beta}^k) \mathbf{J}(\mathbf{r}, \hat{\beta}^k)$. See Berndt et al. (1974), Fletcher and Powell (1963) and Kennedy and Gentle (1980) for some choices of α_{k+1} .

Gauss–Newton Method

The Gauss–Newton method is designed specifically for NLS by replacing the regression function $h(x_i, \beta)$ in (8.17) by its first-order Taylor expansion. The resulting iteration step is

$$\hat{\beta}^{k+1} = \hat{\beta}^k - \left\{ \mathbf{J}(\mathbf{h}, \hat{\beta}^k)^\top \mathbf{J}(\mathbf{h}, \hat{\beta}^k) \right\}^{-1} \mathbf{J}(\mathbf{h}, \hat{\beta}^k)^\top \mathbf{r}(\hat{\beta}^k). \quad (8.20)$$

Being rather similar to Newton's method, it does not require the Hessian matrix $H(r^2, \hat{\beta}^k)$, which is "approximated" by $J(\mathbf{h}, \hat{\beta}^k)^\top J(\mathbf{h}, \hat{\beta}^k)$ (both matrices are equal in probability for $n \rightarrow \infty$ under assumptions of Theorem 5, see Amemiya, 1985). Because it only approximates the true Hessian matrix, this method belongs to the class of quasi-Newton methods. The issues discussed in the case of Newton's method apply also to the Gauss–Newton method.

Levenberg–Marquardt Method

Depending on data and the current approximation $\hat{\beta}^k$ of $\hat{\beta}^{\text{NLS}}$, the Hessian matrix $H(\hat{\beta}^k)$ or its approximations such as $J(\mathbf{h}, \hat{\beta}^k)^\top J(\mathbf{h}, \hat{\beta}^k)$ can be badly conditioned or not positive definite, which could even result in divergence of Newton's method (or a very slow convergence in the case of modified Newton's method). The Levenberg–Marquardt method addresses the ill-conditioning by choosing the search direction $\mathbf{d}_k = \hat{\beta}^{\text{NLS}} - \hat{\beta}^k$ as a solution of

$$\{J(\mathbf{h}, \hat{\beta}^k)^\top J(\mathbf{h}, \hat{\beta}^k) + \tau I_p\} \mathbf{d}_k = -J(\mathbf{h}, \hat{\beta}^k)^\top \mathbf{r}(\hat{\beta}^k) \quad (8.21)$$

(see Marquardt, 1963). This approach is an analogy of RR used in linear regression (Sect. 8.1.6). Similarly to RR, the Levenberg–Marquardt method improves conditioning of the Hessian matrix and it limits the length of the innovation vector \mathbf{d}_k compared to the (Gauss-)Newton method. See Kennedy and Gentle (1980) and Björck (1996) for a detailed discussion of this algorithm. There are also algorithms combining both Newton's and the Levenberg–Marquardt approaches by using at each step the method that generates a larger reduction in objective function.

Although Newton's method and its modifications are most frequently used in applications, the fact that they find local minima gives rise to various improvements and alternative methods. They range from simple starting the minimization algorithm from several (randomly chosen) initial points to general global-search optimization methods such as genetic algorithms mentioned in Sect. 8.1.3 and discussed in more details in Chaps. II.5 and II.6.

Statistical Inference

8.2.2

Similarly to linear modeling, the inference in nonlinear regression models is mainly based, besides the estimate $\hat{\beta}^{\text{NLS}}$ itself, on two quantities: the residual sum of squares $\text{RSS} = \mathbf{r}(\hat{\beta}^{\text{NLS}})^\top \mathbf{r}(\hat{\beta}^{\text{NLS}})$ and the (asymptotic) variance of the estimate $\text{Var}(\hat{\beta}^{\text{NLS}}) = \sigma^2 \mathbf{A}(\beta_0)^{-1}$, see (8.18). Here we discuss how to compute these quantities for $\hat{\beta}^{\text{NLS}}$ and its functions.

RSS will be typically a by-product of a numerical computation procedure, since it constitutes the minimized function. RSS also provides an estimate of σ^2 : $s^2 = \text{RSS}/(n - k)$. The same also holds for the matrix $\mathbf{A}(\beta_0)$, which can be consistently estimated by $\mathbf{A}(\hat{\beta}^{\text{NLS}}) = J(\mathbf{h}, \hat{\beta}^k)^\top J(\mathbf{h}, \hat{\beta}^k)$, that is, by the asymptotic representation of the Hessian matrix $H(r^2, \hat{\beta}^k)$. This matrix or its approximations are computed at every step of (quasi-)Newton methods for NLS, and thus, it will be readily available after the estimation.

Furthermore, the inference in nonlinear regression models may often involve a nonlinear (vector) function of the estimate $f(\hat{\boldsymbol{\beta}}^{\text{NLS}})$; for example, when we test a hypothesis (see Amemiya, 1983, for a discussion of NLS hypothesis testing). Contrary to linear functions of estimates, where $\text{Var}(A\hat{\boldsymbol{\beta}}^{\text{NLS}} + \mathbf{a}) = A^\top \text{Var}(\hat{\boldsymbol{\beta}}^{\text{NLS}})A$, there is no exact expression for $\text{Var}[f(\hat{\boldsymbol{\beta}}^{\text{NLS}})]$ in a general case. Thus, we usually assume the first-order differentiability of $f(\cdot)$ and use the Taylor expansion to approximate this variance. Since

$$f(\hat{\boldsymbol{\beta}}) = f(\boldsymbol{\beta}_0) + \frac{\partial f(\boldsymbol{\beta}_0)}{\partial \boldsymbol{\beta}^\top} (\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}_0) + o(\|\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}_0\|),$$

it follows that the variance can be approximated by

$$\text{Var}[f(\hat{\boldsymbol{\beta}}^{\text{NLS}})] \doteq \frac{\partial f(\hat{\boldsymbol{\beta}}^{\text{NLS}})}{\partial \boldsymbol{\beta}^\top} \text{Var}(\hat{\boldsymbol{\beta}}^{\text{NLS}}) \frac{\partial f(\hat{\boldsymbol{\beta}}^{\text{NLS}})}{\partial \boldsymbol{\beta}}.$$

Hence, having an estimate of $\text{Var}(\hat{\boldsymbol{\beta}}^{\text{NLS}})$, the Jacobian matrix $\partial f/\partial \boldsymbol{\beta}^\top$ of function f evaluated at $\hat{\boldsymbol{\beta}}^{\text{NLS}}$ provides the first-order approximation of the variance of $f(\hat{\boldsymbol{\beta}}^{\text{NLS}})$.

8.2.3 Ill-conditioned Nonlinear System

Similarly to linear modeling, the nonlinear models can also be ill-conditioned when the Hessian matrix $H(r^2, \boldsymbol{\beta})$ is nearly singular or does not even have a full rank, see Sect. 8.1.2. This can be caused either by the nonlinear regression function h itself or by too many explanatory variables relative to sample size n . Here we mention extensions of methods dealing with ill-conditioned problems in the case of linear models (discussed in Sects. 8.1.5–8.1.9) to nonlinear modeling: ridge regression, Stein-rule estimator, Lasso, and partial least squares.

First, one of early nonlinear RR was proposed by Dagenais (1983), who simply added a diagonal matrix to $\mathbf{H}(r^2, \boldsymbol{\beta})$ in (8.19). Since the nonlinear modeling is done by minimizing of an objective function, a more straightforward way is to use the alternative formulation (8.11) of RR and to minimize

$$\sum_{i=1}^n \{y_i - h(\mathbf{x}_i^\top, \boldsymbol{\beta})\}^2 + k \sum_{j=1}^p \beta_j^2 = \mathbf{r}(\boldsymbol{\beta})^\top \mathbf{r}(\boldsymbol{\beta}) + k \|\boldsymbol{\beta}\|_2^2, \quad (8.22)$$

where k represents the ridge coefficient. See Ngo et al. (2003) for an application of this approach.

Next, equally straightforward is an application of Stein-rule estimator (8.8) in nonlinear regression, see Kim and Hill (1995) for a recent study of positive-part Stein-rule estimator within the Box–Cox model. The same could possibly apply to Lasso-type estimators discussed in Sect. 8.1.8 as well: the Euclidian norm $\|\boldsymbol{\beta}\|_2^2$ in (8.22) would just have to be replaced by another L_q norm. Nevertheless, the

behavior of Lasso within linear regression has only recently been studied in more details, and to my best knowledge, there are no results on Lasso in nonlinear models yet.

Finally, there is a range of modifications of PLS designed for nonlinear regression modeling, which either try to make the relationship between dependent and expl variables linear in unknown parameters or deploy an intrinsically nonlinear model. First, the methods using linearization are typically based on approximating a nonlinear relationship by higher-order polynomials (see quadratic PLS by Wold et al. (1989), and INLR approach by Berglund and Wold (1997)) or a piecewise constant approximation (GIFI approach, see Berglund et al., 2001). Wold et al (2001) present an overview of these methods. Second, several recent works introduced intrinsic nonlinearity into PLS modeling. Among most important contributions, there are Qin and McAvoy (1992) and Malthouse et al. (1997) modeling the nonlinear relationship using a forward-feed neural network, Wold (1992) and Durand and Sabatier (1997) transforming predictors by spline functions, and Bang et al. (2003) using fuzzy-clustering regression approach.

References

- Akaike, H. (1974). A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, 19: 716–723.
- Akdeniz, F., Yüksel, G., and Wan, A.T.K. (2004). The moments of the operational almost unbiased ridge regression estimator. *Applied Mathematics and Computation*: in press.
- Amemiya, T. (1983). Non-linear regression models. In Griliches, Z. and Intriligator, M.D. (eds) *Handbook of Econometrics, Volume 1*. North-Holland Publishing Company, Amsterdam.
- Amemiya, T. (1985). *Advanced Econometrics*. Harvard University Press, Cambridge, USA.
- Bang, Y.H., Yoo, C.K. and Lee, I-B. (2003). Nonlinear PLS modeling with fuzzy inference system. *Chemometrics and Intelligent Laboratory Systems*, 64: 137–155.
- Barlow, J.L. (1993). Numerical aspects of solving linear least squares problems. In Rao, C.R. (ed), *Handbook of Statistics, Volume 9*. Elsevier, Amsterdam London New York Tokyo.
- Barros, A.S. and Rutledge, D.N. (1998). Genetic algorithm applied to the selection of principal components. *Chemometrics and Intelligent Laboratory Systems*, 40: 65–81.
- Bates, D.M. and Watts, D.G. (1988). *Nonlinear Regression Analysis and Its Applications*. Wiley, New York, USA.
- Bedrick, E.J. and Tsai, C-L. (1994). Model Selection for Multivariate Regression in Small Samples. *Biometrics*, 50: 226–231.
- Berglund, A. and Wold, S. (1997). INLR, implicit nonlinear latent variable regression. *Journal of Chemometrics*, 11: 141–156.

- Berglund, A., Kettaneh, Wold, S., Bendwell, N. and Cameron, D.R. (2001). The GIF approach to non-linear PLS modelling. *Journal of Chemometrics*, 15: 321–336.
- Berndt, E.R., Hall, B.H., Hall, R.E. and Hausman, J.A. (1974). Estimation and Inference in Nonlinear Structural Models. *Annals of Econometric and Social Measurement*, 3: 653–666
- Björck, A. (1996). *Numerical Methods for Least Squares Problems*. SIAM Press, Philadelphia, USA.
- Björkström, A. and Sundberg, R. (1996). Continuum regression is not always continuous. *Journal of Royal Statistical Society B*, 58: 703–710.
- Björkström, A. and Sundberg, R. (1999). A generalized view on continuum regression. *Scandinavian Journal of Statistics*, 26: 17–30.
- Brooks, R. and Stone, M. (1990). Continuum regression: cross-validated sequentially constructed prediction embracing ordinary least squares, partial least squares and principal component regression. *Journal of Royal Statistical Society B*, 52: 237–269.
- Brooks, R. and Stone, M. (1994). Joint continuum regression for multiple predictants. *Journal of American Statistical Association*, 89: 1374–1377.
- Chambers, L. (1998). *Practical Handbook of Genetic Algorithms: Complex Coding Systems, Volume III*. CRC Press, USA.
- Chawla, J.S. (1990). A note on ridge regression. *Statistics & Probability Letters*, 9: 343–345.
- Coutis, C. (1996). Partial least squares algorithm yields shrinkage estimators. *The Annals of Statistics*, 24: 816–824.
- Dagenais, M.G. (1983). Extension of the ridge regression technique to non-linear models with additive errors. *Economic Letters*, 12: 169–174.
- Danilov, D. and Magnus, J.R. (2004). On the harm that ignoring pretesting can cause. *Journal of Econometrics*, in press.
- Denham, M.C. (1997). Prediction intervals in partial least squares. *Journal of Chemometrics*, 11: 39–52
- Depczynski, U., Frost, V.J. and Molt, K. (2000). Genetic algorithms applied to the selection of factors in principal component regression. *Analytica Chimica Acta*, 420: 217–227.
- Durand, J-F. and Sabatier, R. (1997). Additive spline for partial least squares regression. *Journal of American Statistical Association*, 92: 1546–1554.
- Edwards, D. and Havranek, T. (1987). A fast model selection procedure for large families of models. *Journal of American Statistical Association*, 82: 205–213.
- Efron, B., Hastie, T., Johnstone, I. and Tibshirani, R. (2004). Least Angle Regression. *Annals of Statistics*, 32: in press.
- Efroymson, M.A. (1960). Multiple regression analysis. In Ralston, A. and Wilf, H.S. (eds), *Mathematical Methods for Digital Computers, Vol. 1*, Wiley, New York, USA.
- Faber, N.M., Song, X-H. and Hopke, P.K. (2003). Sample specific standard error of prediction for partial least squares regression. *Trends in Analytical Chemistry*, 22: 330–334.

- Farebrother, R.W. (1976). Further results on the mean square error of ridge estimation. *Journal of Royal Statistical Society B*, 38: 248–250.
- Fletcher, R. and Powell, M.J.D. (1963). A rapidly convergent descent method for minimization. *Computer Journal* 6: 163–168.
- Frank, I.E., Friedman, J.H., Wold, S., Hastie, T. and Mallows, C. (1993). A statistical view of some chemometrics regression tools. *Technometrics*, 35(2): 109–148.
- Garthwaite, P.H. (1994). An interpretation of partial least squares. *The Journal of American Statistical Association*, 89: 122–127.
- Gentle, J.E. (1998). *Numerical Linear Algebra for Applications in Statistics*. Springer, New York, USA.
- Gruber, M.H.J. (1998). *Improving efficiency by shrinkage: the James-Stein and ridge regression estimators*. Marcel Dekker, Inc., New York, USA.
- Gunst, R.F. and Mason, R.L. (1980). *Regression Analysis and Its Application: a Data-Oriented Approach*. Marcel Dekker, Inc., New York, USA.
- Härdle, W. (1992). *Applied Nonparametric Regression*. Cambridge University Press, Cambridge, UK.
- Härdle, W. and Simar, L. (2003). *Applied Multivariate Statistical Analysis*. Springer, Heidelberg, Germany.
- Hawkins, D.M. and Yin, X. (2002). A faster algorithm for ridge regression of reduced rank data. *Computational Statistics & Data analysis*, 40: 253–262.
- Helland, I.S. (2001). Some theoretical aspects of partial least squares regression. *Chemometrics and Intelligent Laboratory Systems*, 58: 97–107.
- Helland, I.S. and Almoy, T. (1994). Comparison of Prediction Methods When Only a Few Components are Relevant. *Journal of American Statistical Association*, 89: 583–591.
- Hocking, R.R. (1996). *Methods and Applications of Linear Models: Regression and the Analysis of Variance, 2nd Edition*. Wiley, New York, USA.
- Hoerl, A.E. and Kennard, R.W. (1970). Ridge regression: biased estimation of nonorthogonal problems. *Technometrics*, 12: 55–67.
- Hoerl, A.E., Kennard, R.W. and Baldwin, K.F. (1975). Ridge regression: some simulations. *Communications in Statistics*, 4: 105–123.
- Hughes, A.W. and Maxwell, L.K. (2003). Model selection using AIC in the presence of one-sided information. *Journal of Statistical Planning and Inference*, 115: 379–411.
- Hwang, J.T.G. and Nettleton, D. (2003). Principal components regression with data-chosen components and related methods. *Technometrics*, 45: 70–79.
- Ibrahim, J.G. and Ming-Hui, C. (1997). Predictive Variable Selection for the Multivariate Linear Model. *Biometrics*, 53: 465–478.
- Jiang, W. and Liu, X. (2004). Consistent model selection based on parameter estimates. *Journal of Statistical Planning and Inference*, 121: 265–283.
- Jolliffe, I.T. (1982). A note on the use of the principle components in regression. *Applied Statistics*, 31(3): 300–303.
- Jong, S. (1993). SIMPLS: An alternative approach to partial least squares regression. *Chemometrics and Intelligent Laboratory Systems*, 18: 251–263.

- Jong, S. (1995). PLS shrinks. *Journal of Chemometrics*, 9: 323–326.
- Judge, G.G. and Bock, M.E. (1983). Biased estimation. In Griliches, Z. and Intriligator, M.D. (eds), *Handbook of Econometrics, Volume 1*, North-Holland Publishing Company, Amsterdam.
- Kadiyala, K. (1984). A class of almost unbiased and efficient estimators of regression coefficients. *Economic Letters*, 16: 293–296.
- Kennedy, W. J. and Gentle, J.E. (1980). *Statistical Computing*. Marcel Dekker, Inc., New York, USA.
- Kibria, G. (1996). On preliminary test ridge regression estimators for linear restrictions in a regression model with non-normal disturbances. *Communications in Statistics, Theory and Methods*, 25: 2349–2369.
- Kim, M. and Hill, R.C. (1995). Shrinkage estimation in nonlinear regression: the Box-Cox transformation. *Journal of Econometrics*, 66: 1–33.
- Knight, K. and Fu, W. (2000). Asymptotics for Lasso-type estimators. *The Annals of Statistics*, 28: 1356–1389.
- Leardi, R. and Gonzáles, A.L. (1998). Genetic algorithms applied to feature selection in PLS regression: how and when to use them. *Chemometrics and Intelligent Laboratory Systems*, 41: 195–207.
- Leamer, E.E. (1983). Model choice and specification analysis. In Griliches, Z. and Intriligator, M.D. (eds), *Handbook of Econometrics, Volume 1*, North-Holland Publishing Company, Amsterdam.
- Li, K-C. (1987). Asymptotic optimality for C_p , C_L , cross-validation and generalized cross-validation: discrete index set. *Annals of Statistics*, 15: 958–975.
- Li, B., Morris, J. and Martin, E.B. (2002). Model selection for partial least squares regression. *Chemometrics and Intelligent Laboratory Systems*, 64: 79–89.
- Magnus, J.R. (1999). The traditional pretest estimator. *Theory of Probability and Its Applications*. 44(2): 293–308.
- Magnus, J.R. (2002). Estimation of the mean of a univariate normal distribution with known variance. *The Econometrics Journal*, 5, 225–236.
- Malthouse, E.C., Tamhane, A.C. and Mah, R.S.H. (1997). Nonlinear partial least squares. *Computers in Chemical Engineering*, 21(8): 875–890.
- Marquardt, D.W. (1963). An algorithm for least-squares estimation of nonlinear parameters. *Journal of the Society for Industrial and Applied Mathematics*, 11: 431–441.
- McDonald, G.C. and Schwing, R.C. (1973). Instabilities of regression estimates relating air pollution to mortality. *Technometrics*, 15: 463–482.
- Miller, A.J. (1984). Selection of Subsets of Regression Variables. *Journal of the Royal Statistical Society A*, 147(3): 389–425.
- Miller, A. (2002). *Subset Selection in Regression*, Chapman & Hall/CRC, USA.
- Montgomery, D.C., Peck, E.A. and Vining, G.G. (2001). *Introduction to Linear Regression Analysis, 3rd Edition*, Wiley, New York, USA.
- Ngo, S.H., Kemény, S. and Deák, A. (2003). Performance of the ridge regression methods as applied to complex linear and nonlinear models. *Chemometrics and Intelligent Laboratory Systems*, 67: 69–78.

- Ohtani, K. (1986). On small sample properties of the almost unbiased generalized ridge estimator. *Communications in Statistics, Theory and Methods*, 22: 2733–2746.
- Osborne, M.R., Presnell, B. and Turlach, B.A. (1999). On the Lasso and its dual. *Journal of Computational and Graphical Statistics*, 9: 319–337.
- Osten, D.W. (1988). Selection of optimal regression models via cross-validation. *Journal of Chemometrics*, 2: 39–48.
- Phatak, A., Reilly, P.M. and Pendilis, A. (2002). The asymptotic variance of the univariate PLS estimator. *Linear Algebra and its Applications*, 354: 245–253.
- Rao, C.R. and Toutenberg, H. (1999). *Linear Models*, Springer, New York, USA.
- Rao, C.R. and Wu, Y. (1989). A strongly consistent procedure for model selection in a regression problem. *Biometrika*, 76: 369–374.
- Qin, S. and McAvoy, T. (1992). Nonlinear PLS modeling using neural networks. *Computers in Chemical Engineering*, 16: 379–391.
- Qin, S.J. (1997). Recursive PLS algorithms for adaptive data modeling. *Computers in Chemical Engineering*, 22(4): 503–514.
- Schwarz, G. (1978). Estimating the dimension of a model. *The Annals of Statistics*, 6: 461–464.
- Seber, G.A.F. and Wild, C.J. (2003). *Nonlinear Regression*, Wiley, New York, USA.
- Shao, J. (1993). Linear model selection by cross-validation. *Journal of American Statistical Association*, 88: 486–494.
- Shao, J. (1997). An asymptotic theory for linear model selection. *Statistica Sinica*, 7: 221–264.
- Shi, P. and Tsai, C.-L. (1998). A note on the unification of the Akaike information criterion. *Journal of the Royal Statistical Society B*, 60: 551–558.
- Shiaishi, T. and Konno, Y. (1995). On construction of improved estimators in multiple-design multivariate linear models under general restrictions. *Annals of Institute of Statistical Mathematics*, 46: 665–674.
- Shibata, R. (1981). An optimal selection of regression variables. *Biometrika*, 68: 45–54.
- Shibata, R. (1984). Approximate efficiency of a selection procedure for the number of regression variables. *Biometrika*, 71: 43–49.
- Singh, R.K., Pandey, S. K. and Srivastava, V.K. (1994). A generalized class of shrinkage estimators in linear regression when disturbances are not normal. *Communications in Statistics, Theory and Methods*, 23: 2029–2046.
- Stoica, P. and Söderström, T. (1998). Partial least squares: a first-order analysis. *Scandinavian Journal of Statistics*, 25: 17–26.
- Stone, M. (1974). Cross-validatory choice and assessment of statistical predictions. *Journal of Royal Statistical Society B*, 36: 111–147.
- Sundberg, R. (1993). Continuum regression and ridge regression. *Journal of Royal Statistical Society B*, 55: 653–659.
- Swamy, P.A.V.B., Mehta, J.S. and Rappoport, P.N. (1978). Two methods of evaluating Hoerl and Kennard's ridge regression. *Communications in Statistics A*, 12: 1133–1155.

- Thisted, R.A. (1988). *Elements of Statistical Computing*. Chapman and Hall, London New York.
- Tibshirani, R. (1996). Regression shrinkage and selection via Lasso. *Journal of Royal Statistical Society B*, 58: 267–288.
- Trygg, J. and Wold, S. (2002). Orthogonal projections to latent structures, O-PLS. *Journal of Chemometrics*, 16(3): 119–128.
- Ullah, A., Sristava, V.K. and Chandra, R. (1983). Properties of shrinkage estimators in linear regression when disturbances are not normal. *Journal of Econometrics*, 21: 289–402.
- Vinod, H.D. and Ullah, A. (1981). *Recent Advances in Regression Methods*. Marcel Dekker Inc., New York, USA.
- Wan, A.T.K. (2002). On generalized ridge regression estimators under collinearity and balanced loss. *Applies Mathematics and Computation*, 129: 455–467.
- Wang, S.G. and Chow, S.C. (1990). A note on adaptive generalized ridge regression estimator. *Statistics & Probability Letters*, 10: 17–21.
- Wang, S.G., Tse, S.K. and Chow, S.C. (1990). On the measures of multicollinearity in least squares regression. *Statistics & Probability Letters*, 9: 347–355.
- Wasserman, G.S. and Sudjianto, A. (1994). All subsets regression using a generic search algorithm. *Computers and Industrial Engineering*, 27: 489–492.
- Wegelin, J.A. (2000). A survey of partial least squares (PLS) methods, with emphasis on the two-block case. Technical Report 371, Department of Statistics, University of Washington, Seattle.
- Weiss, R.E. (1995). The influence of variable selection: a bayesian diagnostic perspective. *Journal of the American Statistical Association*, 90: 619–625.
- Wentzell, P.D. and Montoto, L.V. (2003). Comparison of principal components regression and partial least squares regression through generic simulations of complex mixtures. *Chemometrics and Intelligent Laboratory Systems*, 65: 257–279.
- Wold, H. (1966). Estimation of principle components and related models by iterative least squares. In Krishnaiah (ed) *Multivariate analysis*. Academic Press, New York.
- Wold, S. (1978). Cross-validation estimation of the number of components in factor and principal components analysis. *Technometrics*, 24: 397–405.
- Wold, S. (1992). Nonlinear partial least squares modelling II. Spline inner relation. *Chemometrics and Intelligent Laboratory Systems*, 14: 71–84.
- Wold, S., Kettaneh-Wold, N. and Skagerberg, B. (1989). Nonlinear PLS modelling. *Chemometrics and Intelligent Laboratory Systems*, 7: 53–65.
- Wold, S., Trygg, J., Berglund, A. and Atti, H. (2001). Some recent developments in PLS modeling. *Chemometrics and Intelligent Laboratory Systems*, 58: 131–150.
- Zhang, P. (1992). Inference after variable selection in linear regression models. *Biometrika*, 79(4): 741–746.
- Zheng, X. and Loh, W-Y. (1995). Consistent variable selection in linear models. *Journal of the American Statistical Association*, 90: 151–156.

Robust Statistics

III.9

Laurie Davies, Ursula Gather

9.1	<i>Robust Statistics; Examples and Introduction</i>	656
	Two Examples	656
	General Philosophy	657
	Functional Approach	660
9.2	<i>Location and Scale in \mathbb{R}</i>	661
	Location, Scale and Equivariance	661
	Existence and Uniqueness	662
	M-estimators	662
	Bias and Breakdown	665
	Confidence Intervals and Differentiability	668
	Efficiency and Bias	669
	Outliers in \mathbb{R}	670
9.3	<i>Location and Scale in \mathbb{R}^k</i>	672
	Equivariance and Metrics	672
	M-estimators of Location and Scale	674
	Bias and Breakdown	675
	High Breakdown Location and Scale Functionals in \mathbb{R}^k	676
	Outliers in \mathbb{R}	679
9.4	<i>Linear Regression</i>	681
	Equivariance and Metrics	681
	M-estimators for Regression	681
	Bias and Breakdown	682
	High Breakdown Regression Functionals	683
	Outliers	684

9.5	<i>Analysis of Variance</i>	685
	One-way Table	685
	Two-way Table	687

Robust Statistics; Examples and Introduction

9.1

Two Examples

9.1.1

The first example involves the real data given in Table 9.1 which are the results of an interlaboratory test. The boxplots are shown in Fig. 9.1 where the dotted line denotes the mean of the observations and the solid line the median.

We note that only the results of the Laboratories 1 and 3 lie below the mean whereas all the remaining laboratories return larger values. In the case of the median, 7 of the readings coincide with the median, 24 readings are smaller and 24 are larger. A glance at Fig. 9.1 suggests that in the absence of further information the Laboratories 1 and 3 should be treated as outliers. This is the course which we recommend although the issues involved require careful thought. For the moment we note simply that the median is a robust statistic whereas the mean is not.

The second example concerns quantifying the scatter of real valued observations x_1, \dots, x_n . This example is partially taken from Huber (1981) and reports a dispute

Table 9.1. The results of an interlaboratory test involving 14 laboratories

1	2	3	4	5	6	7	9	9	10	11	12	13	14
1.4	5.7	2.64	5.5	5.2	5.5	6.1	5.54	6.0	5.1	5.5	5.9	5.5	5.3
1.5	5.8	2.88	5.4	5.7	5.8	6.3	5.47	5.9	5.1	5.5	5.6	5.4	5.3
1.4	5.8	2.42	5.1	5.9	5.3	6.2	5.48	6.1	5.1	5.5	5.7	5.5	5.4
0.9	5.7	2.62	5.3	5.6	5.3	6.1	5.51	5.9	5.3	5.3	5.6	5.6	

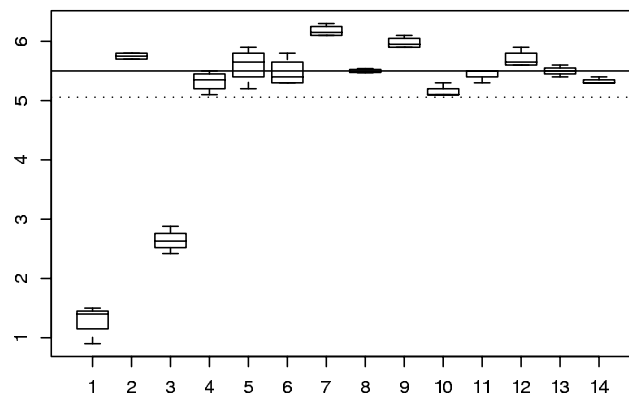


Figure 9.1. A boxplot of the data of Table 9.1. The dotted line and the solid line denote respectively the mean and the median of the observations

between Eddington (1914), p. 147 and Fisher (1920), p. 762 about the relative merits of

$$s_n = \left(\frac{1}{n} \sum (x_i - \bar{x})^2 \right)^{\frac{1}{2}} \quad \text{and} \quad d_n = \frac{1}{n} \sum |x_i - \bar{x}|.$$

Fisher argued that for normal observations the standard deviation s_n is about 12% more efficient than the mean absolute deviation d_n . In contrast Eddington claimed that his experience with real data indicates that d_n is better than s_n . In Tukey (1960) and Huber (1977) we find a resolution of this apparent contradiction. Consider the model

$$\mathcal{N}_\varepsilon = (1 - \varepsilon)N(\mu, \sigma^2) + \varepsilon N(\mu, 9\sigma^2), \quad (9.1)$$

where $N(\mu, \sigma^2)$ denotes a normal distribution with mean μ and variance σ^2 and $0 \leq \varepsilon \leq 1$. For data distributed according to (9.1) one can calculate the asymptotic relative efficiency ARE of d_n with respect to s_n ,

$$\text{ARE}(\varepsilon) = \lim_{n \rightarrow \infty} \text{RE}_n(\varepsilon) = \lim_{n \rightarrow \infty} \frac{\text{Var}(s_n)/E(s_n)^2}{\text{Var}(d_n)/E(d_n)^2}.$$

As Huber states, the result is disquieting. Already for $\varepsilon \geq 0.002$ ARE exceeds 1 and the effect is apparent for samples of size 1000. For $\varepsilon = 0.05$ we have $\text{ARE}(\varepsilon) = 2.035$ and simulations show that for samples of size 20 the relative efficiency exceeds 1.5 and increases to 2.0 for samples of size 100. This is a severe deficiency of s_n as models such as \mathcal{N}_ε with ε between 0.01 and 0.1 often give better descriptions of real data than the normal distribution itself. We quote Huber (1981)

thus it becomes painfully clear that the naturally occurring deviations from the idealized model are large enough to render meaningless the traditional asymptotic optimality theory.

9.1.2 General Philosophy

The two examples of the previous section illustrate a general phenomenon. An optimal statistical procedure based on a particular family of models \mathcal{M}_1 can differ considerably from an optimal procedure based on another family \mathcal{M}_2 even though the families \mathcal{M}_1 and \mathcal{M}_2 are very close. This may be expressed by saying that optimal procedures are often unstable in that small changes in the data or the model can lead to large changes in the analysis. The basic philosophy of robust statistics is to produce statistical procedures which are stable with respect to small changes in the data or model and even large changes should not cause a complete breakdown of the procedure.

Any inspection of the data and the removal of aberrant observations may be regarded as part of robust statistics but it was only with Pearson (1931) that the consideration of deviations from models commenced. He showed that the ex-

act theory based on the normal distribution for variances is highly nonrobust. There were other isolated papers on the problem of robustness (Pearson (1929); Bartlett (1935); Geary (1936, 1937); Gayen (1950); Box (1953); Box and Andersen (1955)). Tukey (1960) initiated a wide spread interest in robust statistics which has continued to this day. The first systematic investigation of robustness is due to Huber (1964) and was expounded in Huber (1981). Huber's approach is functional analytic and he was the first to investigate the behaviour of a statistical functional over a full topological neighbourhood of a model instead of restricting the investigation to other parametric families as in (9.1). Huber considers three problems. The first is that of minimizing the bias over certain neighbourhoods and results in the median as the most robust location functional. For large samples deviations from the model have consequences which are dominated by the bias and so this is an important result. The second problem is concerned with what Tukey calls the statistical version of no free lunches. If we take the simple model of i.i.d. $N(\mu, 1)$ observations then the confidence interval for μ based on the mean is on average shorter than that based on any other statistic. If short confidence intervals are of interest then one can not only choose the statistic which gives the shortest interval but also the model itself. The new model must of course still be consistent with the data but even with this restriction the confidence interval can be made as small as desired (Davies (1995)). Such a short confidence interval represents a free lunch and if we do not believe in free lunches then we must look for that model which *maximizes* the length of the confidence interval over a given family of models. If we take all distributions with variance 1 then the confidence interval for the $N(\mu, 1)$ distribution is the longest. Huber considers the same problem over the family $\mathcal{F} = \{F : d_{ko}(F, N(0, 1)) < \varepsilon\}$ where d_{ko} denotes the Kolmogoroff metric. Under certain simplifying assumptions Huber solves this problem and the solution is known as the Huber distribution (see Huber (1981)). Huber's third problem is the robustification of the Neyman–Pearson test theory. Given two distributions P_0 and P_1 Neyman and Pearson (1933) derive the optimal test for testing P_0 against P_1 . Huber considers full neighbourhoods \mathcal{P}_0 of P_0 and \mathcal{P}_1 of P_1 and then derives the form of the minimax test for the composite hypothesis of \mathcal{P}_0 against \mathcal{P}_1 . The weakness of Huber's approach is that it does not generalize easily to other situations. Nevertheless it is the spirit of this approach which we adopt here. It involves treating estimators as functionals on the space of distributions, investigating where possible their behaviour over full neighbourhoods and always being aware of the danger of a free lunch.

Hampel (1968) introduced another approach to robustness, that based on the influence function $I(x, T, F)$ defined for a statistical functional T as follows

$$I(x, T, F) = \lim_{\varepsilon \rightarrow 0} \frac{T((1 - \varepsilon)F + \varepsilon\delta_x) - T(F)}{\varepsilon}, \quad (9.2)$$

where δ_x denotes the point mass at the point x . The influence function has two interpretations. On the one hand it measures the infinitesimal influence of an observation situated at the point x on the value of the functional T . On the oth-

er hand if $P_n(F)$ denotes the empirical measure of a sample of n i.i.d. random variables with common distribution F then under appropriate regularity conditions

$$\lim_{n \rightarrow \infty} \sqrt{n}(T(P_n(F)) - T(F)) \stackrel{D}{=} N\left(0, \int I(x, T, F)^2 dF(x)\right), \quad (9.3)$$

where $\stackrel{D}{=}$ denotes equality of distribution. Given a parametric family $\mathcal{P}' = \{P_\theta : \theta \in \Theta\}$ of distributions we restrict attention to those functionals which are Fisher consistent that is

$$T(P_\theta) = \theta, \quad \theta \in \Theta. \quad (9.4)$$

Hampel's idea was to minimize the asymptotic variance of T as an estimate of a parameter θ subject to a bound on the influence function

$$\min_T \int I(x, T, P_\theta)^2 dP_\theta(x) \quad \text{under (9.4) and} \quad \sup_x |I(x, T, P_\theta)| \leq k(\theta), \quad (9.5)$$

where $k(\theta)$ is a given function of θ . Hampel complemented the infinitesimal part of his approach by considering also the global behaviour of the functional T . He introduced the concept of breakdown point which has had and continues to have a major influence on research in robust statistics. The approach based on the influence function was carried out in Hampel et al. (1986). The strength of the Hampel approach is that it can be used to robustify in some sense the estimation of parameters in any parametric model. The weaknesses are that (9.5) only bounds infinitesimally small deviations from the model and that the approach does not explicitly take into account the free lunch problem. Hampel is aware of this and recommends simple models but simplicity is an addition to and not an integral part of his approach. The influence function is usually used as a heuristic tool and care must be taken in interpreting the results. For examples of situations where the heuristics go wrong we refer to Davies (1993).

Another approach which lies so to speak between that of Huber and Hampel is the so called shrinking neighbourhood approach. It has been worked out in full generality by Rieder (1994). Instead of considering neighbourhoods of a fixed size (Huber) or only infinitesimal neighbourhoods (Hampel) this approach considers full neighbourhoods of a model but whose size decreases at the rate of $n^{-1/2}$ as the sample size n tends to infinity. The size of the neighbourhoods is governed by the fact that for larger neighbourhoods the bias term is dominant whereas models in smaller neighbourhoods cannot be distinguished. The shrinking neighbourhoods approach has the advantage that it does not need any assumptions of symmetry. The disadvantage is that the size of the neighbourhoods goes to zero so that the resulting theory is only robustness over vanishingly small neighbourhoods.

Functional Approach

Although a statistic based on a data sample may be regarded as a function of the data a more general approach is often useful. Given a data set (x_1, \dots, x_n) we define the corresponding empirical distribution P_n by

$$P_n = \frac{1}{n} \sum_{i=1}^n \delta_{x_i}, \quad (9.6)$$

where δ_x denotes the unit mass in x . Although P_n clearly depends on the sample (x_1, \dots, x_n) we will usually suppress the dependency for the sake of clarity. With this notation we can now regard the arithmetic mean $\bar{x}_n = \sum_{i=1}^n x_i/n$ either as a function of the data or as a function T_{av} of the empirical measure P_n ,

$$\bar{x}_n = \int x dP_n(x) = T_{av}(P_n).$$

The function T_{av} can be extended to all measures P which have a finite mean

$$T_{av}(P) = \int x dP(x), \quad (9.7)$$

and is now a functional defined on a certain subset of the family \mathcal{P} of probability measures on \mathbb{R} . This manner of treating statistics is one whose origins go back to von Mises (1937). In the context of robust statistics it was introduced by Huber (1964) and has proved very useful (see Fernholz (1983)). Another example is given by the functional T_{sh} defined as the length of the shortest interval which carries a mass of at least $1/2$,

$$T_{sh}(P) = \operatorname{argmin}\{|I| : P(I) \geq 1/2, I \subset \mathbb{R}\}, \quad (9.8)$$

where $|I|$ denotes the length of the interval I . The idea of using the shortest half interval goes back to Tukey (see Andrews et al. (1972)) who proposed using the mean of the observations contained in it as a robust location functional.

The space \mathcal{P} may be metricized in many ways but we prefer the Kolmogoroff metric d_{ko} defined by

$$d_{ko}(P, Q) = \sup_{x \in \mathbb{R}} |P((-\infty, x]) - Q((-\infty, x])|. \quad (9.9)$$

The Glivenko–Cantelli theorem states

$$\lim_{n \rightarrow \infty} d_{ko}(P_n(P), P) = 0, \quad \text{a.s.}, \quad (9.10)$$

where $P_n(P)$ denotes the empirical measure of the n random variables $X_1(P), \dots, X_n(P)$ of the i.i.d. sequence $(X_i(P))_1^\infty$. In conjunction with (9.10) the metric d_{ko} makes it possible to connect analytic properties of a functional T and its statistical properties. As a first step we note that a functional T which is locally bounded in the Kolmogoroff metric

$$\sup\{|T(Q) - T(P)| : d_{ko}(P, Q) < \varepsilon\} < \infty, \quad (9.11)$$

for some $\varepsilon > 0$ offers protection against outliers. On moving from local boundedness to continuity we see that if a functional T is continuous at P then the sequence $T(P_n(P))$ is a consistent statistic in that

$$\lim_{n \rightarrow \infty} T(P_n(P)) = T(P), \quad a.s.$$

Finally we consider a functional T which is differentiable at P , that is

$$T(Q) - T(P) = \int I(x, P, T) d(Q - P)(x) + o_P(d_{ko}(P, Q)) \quad (9.12)$$

for some bounded function $I(\cdot, P, T) : \mathbb{R} \rightarrow \mathbb{R}$ where, without loss of generality, $\int I(x, P, T) dP(x) = 0$ (see Clarke (1983)). On putting

$$Q = Q_\varepsilon = (1 - \varepsilon)P + \varepsilon\delta_x$$

it is seen that $I(x, P, T)$ is the influence function of (9.2). As

$$d_{ko}(P_n(P), P) = O_P(1/\sqrt{n}) \quad (9.13)$$

the central limit theorem (9.3) follows immediately. Textbooks which make use of this functional analytic approach are as already mentioned Huber (1981), Hampel et al. (1986), Rieder (1994), and also Staudte and Sheather (1990), a book which can be strongly recommended to students as a well written and at the same time deep introductory text.

9.2 Location and Scale in \mathbb{R}

9.2.1 Location, Scale and Equivariance

Changes in measurement units and baseline correspond to affine transformations on \mathbb{R} . We write

$$\mathcal{A} = \{A : \mathbb{R} \rightarrow \mathbb{R} \text{ with } A(x) = ax + b, a \neq 0, b \in \mathbb{R}\}. \quad (9.14)$$

For any probability measure P and for any $A \in \mathcal{A}$ we define

$$P^A(B) = P(\{x : A(x) \in B\}), \quad B \in \mathcal{B}, \quad (9.15)$$

\mathcal{B} denoting all Borel sets on \mathbb{R} . Consider a subset \mathcal{P}' of \mathcal{P} which is closed under affine transformations, that is

$$P \in \mathcal{P}' \Rightarrow P^A \in \mathcal{P}' \quad \text{for all } P \in \mathcal{P}', A \in \mathcal{A}. \quad (9.16)$$

A functional $T_l : \mathcal{P}' \rightarrow \mathbb{R}$ will be called a location functional on \mathcal{P}' if

$$T_l(P^A) = A(T_l(P)), \quad A \in \mathcal{A}, P \in \mathcal{P}'. \quad (9.17)$$

Similarly we define a functional $T_s : \mathcal{P}' \rightarrow \mathbb{R}_+$ to be a scale functional if

$$T_s(P^A) = |a|T_s(P), \quad A \in \mathcal{A}, A(x) = ax + b, P \in \mathcal{P}'. \quad (9.18)$$

Existence and Uniqueness

9.2.2

The fact that the mean T_{av} of (9.7) cannot be defined for all distributions is an indication of its lack of robustness. More precisely the functional T_{av} is not locally bounded (9.11) in the metric d_{k_0} at any distribution P . The median $\text{MED}(P)$ can be defined at any distribution P as the mid-point of the interval of m -values for which

$$P((-\infty, m]) \geq 1/2 \quad \text{and} \quad P([m, \infty)) \geq 1/2. \quad (9.19)$$

Similar considerations apply to scale functionals. The standard deviation requires the existence of the second moment of a distribution. The median absolute deviation MAD (see Andrews et al. (1972)) of a distribution can be well defined at all distributions as follows. Given P we define P' by

$$P'(B) = P(\{x : |x - \text{MED}(P)| \in B\}), \quad B \in \mathcal{B}.$$

and set

$$\text{MAD}(P) = \text{MED}(P'). \quad (9.20)$$

M-estimators

9.2.3

An important family of statistical functionals is the family of M-functionals introduced by Huber (1964). Let ψ and χ be functions defined on \mathbb{R} with values in the interval $[-1, 1]$. For a given probability distribution P we consider the following two equations for m and s

$$\int \psi\left(\frac{x-m}{s}\right) dP(x) = 0 \quad (9.21)$$

$$\int \chi\left(\frac{x-m}{s}\right) dP(x) = 0. \quad (9.22)$$

If the solution exists and is uniquely defined we denote it by

$$T(P) = (T_1(P), T_s(P)) = (m, s).$$

In order to guarantee existence and uniqueness conditions have to be placed on the functions ψ and χ as well as on the probability measure P . The ones we use are due to Scholz (1971) (see also Huber (1981)) and are as follows:

- ($\psi 1$) $\psi(-x) = -\psi(x)$ for all $x \in \mathbb{R}$.
- ($\psi 2$) ψ is strictly increasing

- ($\psi 3$) $\lim_{x \rightarrow \infty} \psi(x) = 1$
 ($\psi 4$) ψ is continuously differentiable with derivative $\psi^{(1)}$.
- ($\chi 1$) $\chi(-x) = \chi(x)$ for all $x \in \mathbb{R}$.
 ($\chi 2$) $\chi : \mathbb{R}_+ \rightarrow [-1, 1]$ is strictly increasing
 ($\chi 3$) $\chi(0) = -1$
 ($\chi 4$) $\lim_{x \rightarrow \infty} \chi(x) = 1$
 ($\chi 5$) χ is continuously differentiable with derivative $\chi^{(1)}$.
- ($\psi\chi 1$) $\chi^{(1)}/\psi^{(1)} : \mathbb{R}_+ \rightarrow \mathbb{R}_+$ is strictly increasing.

If these conditions hold and P satisfies

$$\Delta(P) = \max_x P(\{x\}) < 1/2 \quad (9.23)$$

then (9.21) and (9.22) have precisely one solution. If we set

$$\mathcal{P}' = \{P : \Delta(P) < 1/2\}$$

then \mathcal{P}' satisfies (9.16) and $T_l : \mathcal{P}' \rightarrow \mathbb{R}$ and $T_s : \mathcal{P}' \rightarrow \mathbb{R}_+$ are a location and a scale functional respectively. Two functions which satisfy the above conditions are

$$\psi(x) = \frac{\exp(x/c) - 1}{\exp(x/c) + 1} \quad (9.24)$$

$$\chi(x) = \frac{x^4 - 1}{x^4 + 1}, \quad (9.25)$$

where $c < 0.39$ is a tuning parameter. The restriction on c is to guarantee ($\psi\chi 1$). Algorithms for calculating the solution of (9.21) and (9.22) are given in the Fortran library ROBETH (Marazzi (1992)) which also contains many other algorithms related to robust statistics.

The main disadvantage of M-functionals defined by (9.21) and (9.22) is ($\psi\chi 1$) which links the location and scale parts in a manner which may not be desirable. In particular there is a conflict between the breakdown behaviour and the efficiency of the M-functional (see below). There are several ways of overcoming this. One is to take the scale function T_s and then to calculate a second location functional by solving

$$\int \bar{\psi} \left(\frac{x - m}{T_s(P)} \right) dP(x) = 0. \quad (9.26)$$

If now $\bar{\psi}$ satisfies ($\psi 1$)–($\psi 4$) then this new functional will exist only under the assumption that the scale functional exists and is non-zero. Furthermore the functional can be made as efficient as desired by a suitable choice of $\bar{\psi}$ removing the conflict between breakdown and efficiency. One possible choice for $T_s(P)$ is

the MAD of (9.20) which is simple, highly robust and which performed well in the Princeton robustness study (Andrews et al. (1972)).

In some situations there is an interest in downweighting outlying observations completely rather than in just bounding their effect. A downweighting to zero is not possible for a ψ -function which satisfies $(\psi 2)$ but can be achieved by using so called redescending ψ -functions such as Tukey's biweight

$$\tilde{\psi}(x) = x(1 - x^2)^2\{|x| \leq 1\}. \tag{9.27}$$

In general there will be many solutions of (9.26) for such ψ -functions and to obtain a well defined functional some choice must be made. One possibility is to take the solution closest to the median, another is to take

$$\operatorname{argmin}_m \int \varphi\left(\frac{x - m}{T_s(P)}\right) dP(x) \tag{9.28}$$

where $\varphi^{(1)} = \tilde{\psi}$. Both solutions pose algorithmic problems. The effect of downweighting outlying observations to zero can be attained by using a so called one-step functional T_{om} defined by

$$T_{om}(P) = T_m(P) + T_s(P) \frac{\int \tilde{\psi}\left(\frac{x - T_m(P)}{T_s(P)}\right) dP(x)}{\int \tilde{\psi}^{(1)}\left(\frac{x - T_m(P)}{T_s(P)}\right) dP(x)} \tag{9.29}$$

where T_m is as above and $\tilde{\psi}$ is redescending. We refer to Hampel et al. (1986) and Rousseeuw and Croux (1994) for more details.

So far all scale functionals have been defined in terms of a deviation from a location functional. This link can be broken as follows. Consider the functional T_{ss} defined to be the solution s of

$$\int \chi\left(\frac{x - y}{s}\right) dP(x) dP(y) = 0, \tag{9.30}$$

where χ satisfies the conditions above. It may be shown that the solution is unique with $0 < s < \infty$, if

$$\sum_{a_i} P(\{a_i\})^2 < 1/4, \tag{9.31}$$

where the a_i denote the countably many atoms of P . The main disadvantage of this method is the computational complexity of (9.30) requiring as it does $O(n^2)$ operations for a sample of size n . If χ is of the form

$$\chi(x) = \begin{cases} a > 0, & |x| > 1 \\ b < 0, & |x| \leq 1 \end{cases},$$

then T_{ss} reduces to a quantile of the $|x_i - x_j|$ and much more efficient algorithms exist which allow the functional to be calculated in $O(n \log n)$ operations (see Croux and Rousseeuw (1992), Rousseeuw and Croux (1992, 1993)).

Although we have defined M -functionals as a solution of (9.21) and (9.22) there are sometimes advantages in defining them as a solution of a minimization problem. Consider the Cauchy distribution with density

$$f(x : \mu, \sigma) = \frac{1}{\pi} \frac{\sigma}{\sigma^2 + (x - \mu)^2}. \quad (9.32)$$

We now define $T_c(P) = (T_{cm}(P), T_{cs}(P))$ by

$$T_c(P) = \operatorname{argmin}_{(m,s)} \left(- \int \log(f(x : m, s)) dP(x) + \frac{1}{2} \log(s) \right). \quad (9.33)$$

This is simply the standard maximum likelihood estimate for a Cauchy distribution but there is no suggestion here that the data are so distributed. If $\Delta(P) < 1/2$ it can be shown that the solution exists and is unique. Moreover there exists a simple convergent algorithm for calculating $(T_{cm}(P), T_{cs}(P))$ for a data sample. We refer to Kent and Tyler (1991) for this and the multidimensional case to be studied below. By differentiating the right hand side of (9.33) it is seen that $(T_{cm}(P), T_{cs}(P))$ may be viewed as an M -functional with a redescending ψ -function.

Another class of functionals defined by a minimization problem is the class of S -functionals. Given a function $\varphi : \mathbb{R} \rightarrow [0, 1]$ which is symmetric, continuous on the right and non-increasing on \mathbb{R}_+ with $\varphi(1) = 1$ and $\lim_{x \rightarrow \infty} \varphi(x) = 0$. We define $(T_{sm}(P), T_{ss}(P))$ by

$$(T_{sm}(P), T_{ss}(P)) = \operatorname{argmin}_{(m,s)} \left\{ s : \int \varphi((x - m)/s) dP(x) \geq 1/2 \right\}. \quad (9.34)$$

A special case is a minor variation of the shortest-half functional of (9.8) which is obtained by taking φ to be the indicator function of the interval $[0, 1)$. Although the existence of solutions of (9.34) is guaranteed if $\Delta(P) < 1/2$ the problem of uniqueness is not trivial and requires the existence of a density subject to certain conditions. If φ is smooth then by differentiation it is seen that $(T_{sm}(P), T_{ss}(P))$ may be regarded as an M -functional with a redescending ψ -function given by $\tilde{\psi} = \varphi^{(1)}$. The minimization problem (9.34) acts as a choice function. We refer to Davies (1987).

9.2.4 Bias and Breakdown

Given a location functional T_l the bias is defined by

$$b(T_l, P, \varepsilon, d_{ko}) = \sup\{|T_l(Q) - T_l(P)| : d_{ko}(P, Q) < \varepsilon\}, \quad (9.35)$$

where by convention $T_l(Q) = \infty$ if T_l is not defined at Q . For a scale functional T_s we set

$$b(T_s, P, \varepsilon, d_{ko}) = \sup\{|\log(T_s(Q)/T_s(P))| : d_{ko}(P, Q) < \varepsilon\}, \quad (9.36)$$

where again by convention $T_s(Q) = \infty$ if T_s is not defined at Q . A popular although weaker form of bias function based on the so called gross error neighbourhood is given by

$$b(T_l, P, \varepsilon, GE) = \sup\{|T_l(Q) - T_l(P)| : Q = (1 - \varepsilon)P + \varepsilon H, H \in \mathcal{P}\} \quad (9.37)$$

with a corresponding definition for $b(T_s, P, \varepsilon, GE)$. We have

$$b(T_l, P, \varepsilon, GE) \leq b(T_l, P, \varepsilon, d_{ko}). \quad (9.38)$$

We refer to Huber (1981) for more details.

The breakdown point $\varepsilon^*(T_l, P, d_{ko})$ of T_l at P with respect to d_{ko} is defined by

$$\varepsilon^*(T_l, P, d_{ko}) = \sup\{\varepsilon : b(T_l, P, \varepsilon, d_{ko}) < \infty\} \quad (9.39)$$

with the corresponding definitions for scale functionals and the gross error neighbourhood. Corresponding to (9.38) we have

$$\varepsilon^*(T_l, P, d_{ko}) \leq \varepsilon^*(T_l, P, GE). \quad (9.40)$$

If a functional T_l has a positive breakdown point at a distribution P then it exhibits a certain degree of stability in a neighbourhood of P as may be seen as follows. Consider a sample x_1, \dots, x_n and add to it k further observations x_{n+1}, \dots, x_{n+k} . If P_n and P_{n+k} denote the empirical measures based on x_1, \dots, x_n and x_1, \dots, x_{n+k} respectively then $d_{ko}(P_n, P_{n+k}) \leq k/(n+k)$. In particular if $k/(n+k) < \varepsilon^*(T_l, P_n, d_{ko})$ then it follows that $T_l(P_{n+k})$ remains bounded whatever the added observations. This finite sample concept of breakdown was introduced by Donoho and Huber (1983). Another version replaces k observations by other values instead of adding k observations and is as follows. Let x_1^k, \dots, x_n^k denote a sample differing from x_1, \dots, x_n in at most k readings. We denote the empirical distributions by P_n^k and define

$$\varepsilon^*(T_l, P_n, fsbp) = \max\{k/n : |T_l(P_n^k)| < \infty\}, \quad (9.41)$$

where P_n^k ranges over all possible x_1^k, \dots, x_n^k . This version of the finite sample breakdown point is called the replacement version as k of the original observations can be replaced by arbitrary values. The two breakdown points are related (see Zuo, 2001). There are corresponding versions for scale functionals.

For location and scale functionals there exist upper bounds for the breakdown points. For location functionals T_l we have

Theorem 1

1

$$\varepsilon^*(T_l, P, d_{ko}) \leq 1/2, \quad (9.42)$$

$$\varepsilon^*(T_l, P, GE) \leq 1/2, \quad (9.43)$$

$$\varepsilon^*(T_l, P_n, fsbp) \leq \lfloor n/2 \rfloor / n. \quad (9.44)$$

We refer to Huber (1981). It may be shown that all breakdown points of the mean are zero whereas the median attains the highest possible breakdown point in each case. The corresponding result for scale functionals is more complicated. Whereas we know of no reasonable metric in (9.42) of Theorem 1 which leads to a different upper bound this is not the case for scale functionals. Huber (1981) shows that for the Kolmogoroff metric d_{ko} the corresponding upper bound is $1/4$ but is $1/2$ for the gross error neighbourhood. If we replace the Kolmogoroff metric d_{ko} by the standard Kuiper metric d_{ku} defined by

$$d_{ku}(P, Q) = \sup\{|P(I) - Q(I)| : I \text{ an interval}\} \quad (9.45)$$

then we again obtain an upper bound of $1/2$. For scale functionals T_s we have

2

Theorem 2

$$\varepsilon^*(T_s, P, d_{ku}) \leq (1 - \Delta(P))/2, \quad (9.46)$$

$$\varepsilon^*(T_s, P, GE) \leq (1 - \Delta(P))/2, \quad (9.47)$$

$$\varepsilon^*(T_s, P_n, fsbp) \leq (1 - \Delta(P))/2. \quad (9.48)$$

Similarly all breakdown points of the standard deviation are zero but, in contrast to the median, the MAD does not attain the upper bounds of (9.44). We have

$$\varepsilon^*(MAD, P_n, fsbp) = \max\{0, 1/2 - \Delta(P_n)\}.$$

A simple modification of the MAD, namely

$$MMAD(P) = \min\{|I| : \tilde{P}(I) \geq (1 + \Delta(I))/2\}, \quad (9.49)$$

where $\tilde{P}(B) = P(\{x : |x - \text{MED}(P)| \in B\})$ and $\Delta(I) = \max\{P(\{x\}), x \in I\}$ can be shown to obtain the highest possible finite sample breakdown point of (9.48).

The M-functional defined by (9.21) and (9.22) has a breakdown point ε^* which satisfies

$$\psi^{-1}\left(\frac{\varepsilon^*}{1 - \varepsilon^*}\right) = \chi^{-1}\left(\frac{-\varepsilon^*}{1 - \varepsilon^*}\right) \quad (9.50)$$

(see Huber (1981)). For the functions defined by (9.24) and (9.25) the breakdown point is a decreasing function of c . As c tends to zero the breakdown point tends to $1/2$. Indeed, as c tends to zero the location part of the functional tends to the median. For $c = 0.2$ numerical calculations show that the breakdown point is 0.48. The calculation of breakdown points is not always simple. We refer to Huber (1981) and Gather and Hilker (1997).

The breakdown point is a simple but often effective measure of the robustness of a statistical functional. It does not however take into account the size of the bias. This can be done by trying to quantify the minimum bias over some neighbourhood

of the distribution P and if possible to identify a functional which attains it. We formulate this for $P = N(0, 1)$ and consider the Kolmogoroff ball of radius ε . We have (Huber (1981))

Theorem 3 For every $\varepsilon < 1/2$ we have

3

$$b(\text{MED}, P, \varepsilon, d_{ko}) \leq b(T_l, P, \varepsilon, d_{ko})$$

for any translation functional T_l .

In other words the median minimizes the bias over any Kolmogoroff neighbourhood of the normal distribution. This theorem can be extended to other symmetric distributions and to other situations (Riedel, 1989a, 1989b). It is more difficult to obtain such a theorem for scale functionals because of the lack of a property equivalent to symmetry for location. Nevertheless some results in this direction have been obtained and indicate that the length of the shortest half T_{sh} of (9.8) has very good bias properties (Martin and Zamar (1993b)).

Confidence Intervals and Differentiability

9.2.5

Given a sample x_1, \dots, x_n with empirical measure P_n we can calculate a location functional $T_l(P_n)$ which in some sense describes the location of the sample. Such a point value is rarely sufficient and in general should be supplemented by a confidence interval, that is a range of values consistent with the data. If T_l is differentiable (9.12) and the data are i.i.d. random variables with distribution P then it follows from (9.3) (see Sect. 9.1.3) that an asymptotic α -confidence interval for $T_l(P)$ is given by

$$\left[T_l(P_n(P)) - z((1 + \alpha)/2)\Sigma(P)/\sqrt{n}, T_l(P_n(P)) + z((1 + \alpha)/2)\Sigma(P)/\sqrt{n} \right]. \quad (9.51)$$

Here $z(\alpha)$ denotes the α -quantile of the standard normal distribution and

$$\Sigma(P)^2 = \int I(x, T_l, P)^2 dP(x). \quad (9.52)$$

At first glance this cannot lead to a confidence interval as P is unknown. If however $\Sigma(P)$ is also Fréchet differentiable at P then we can replace $\Sigma(P)$ by $\Sigma(P_n(P))$ with an error of order $O_P(1/\sqrt{n})$. This leads to the asymptotic α -confidence interval

$$\left[T_l(P_n(P)) - z((1 + \alpha)/2)\Sigma(P_n(P))/\sqrt{n}, T_l(P_n(P)) + z((1 + \alpha)/2)\Sigma(P_n(P))/\sqrt{n} \right]. \quad (9.53)$$

A second problem is that (9.53) depends on asymptotic normality and the accuracy of the interval in turn will depend on the rate of convergence to the normal distribution which in turn may depend on P . Both problems can be overcome if T_l is locally uniformly Fréchet differentiable at P . If we consider the M-functionals of Sect. 9.2.3 then they are locally uniformly Fréchet differentiable if the ψ - and

χ -functions are sufficiently smooth (see Bednarski et al. (1991), Bednarski (1993), Bednarski and Clarke (1998), and Davies (1998)). The influence function $I(\cdot, T_l, P)$ is given by

$$I(x, T_l, P) = T_s(P) \frac{D(P) \tilde{\psi} \left(\frac{x - T_l(P)}{T_s(P)} \right) - B(P) \chi \left(\frac{x - T_l(P)}{T_s(P)} \right)}{A(P)D(P) - B(P)C(P)}, \quad (9.54)$$

where

$$A(P) = \int \tilde{\psi}^{(1)} \left(\frac{x - T_l(P)}{T_s(P)} \right) dP(x) \quad (9.55)$$

$$B(P) = \int \left(\frac{x - T_l(P)}{T_s(P)} \right) \tilde{\psi}^{(1)} \left(\frac{x - T_l(P)}{T_s(P)} \right) dP(x) \quad (9.56)$$

$$C(P) = \int \chi^{(1)} \left(\frac{x - T_l(P)}{T_s(P)} \right) dP(x) \quad (9.57)$$

$$D(P) = \int \left(\frac{x - T_l(P)}{T_s(P)} \right) \chi^{(1)} \left(\frac{x - T_l(P)}{T_s(P)} \right) dP(x). \quad (9.58)$$

Simulations suggest that the covering probabilities of the confidence interval (9.53) are good for sample sizes of 20 or more as long as the distribution P is almost symmetric. For the sample x_1, \dots, x_n this leads to the interval

$$\left[T_l(P_n) - z((1 + \alpha)/2)\Sigma(P_n)/\sqrt{n}, T_l(P_n) + z((1 + \alpha)/2)\Sigma(P_n)/\sqrt{n} \right] \quad (9.59)$$

with $\Sigma(P)$ given by (9.52) and $I(x, T_l, P)$ by (9.54). Similar intervals can be obtained for the variations on M-functionals discussed in Sect. 9.2.3.

9.2.6 Efficiency and Bias

The precision of the functional T at the distribution P can be quantified by the length $2z((1 + \alpha)/2)\Sigma(P)/\sqrt{n}$ of the asymptotic confidence interval (9.51). As the only quantity which depends on T is $\Sigma(P)$ we see that an increase in precision is equivalent to reducing the size of $\Sigma(P)$. The question which naturally arises is then that of determining how small $\Sigma(P)$ can be made. A statistical functional which attains this lower bound is asymptotically optimal and if we denote this lower bound by $\Sigma_{\text{opt}}(P)$, the efficiency of the functional T can be defined as $\Sigma_{\text{opt}}(P)^2/\Sigma(P)^2$. The efficiency depends on P and we must now decide which P or indeed P s to choose. The arguments given in Sect. 9.1.2 suggest choosing a P which maximizes $\Sigma_{\text{opt}}(P)$ over a class of models. This holds for the normal distribution which maximizes $\Sigma_{\text{opt}}(P)$ over the class of all distributions with a given variance. For this reason and for simplicity and familiarity we shall take the normal distribution as the reference distribution. If a reference distribution is required which also produces outliers then the slash distribution is to be preferred to the Cauchy distribution. We refer to Cohen (1991) and the discussion given there.

If we consider the M-functionals defined by (9.24) and (9.25) the efficiency at the normal distribution is an increasing function of the tuning parameter c . As the breakdown point is a decreasing function of c this would seem to indicate that there is a conflict between efficiency and breakdown point. This is the case for the M-functional defined by (9.24) and (9.25) and is due to the linking of the location and scale parts of the functional. If this is severed by, for example, recalculating a location functional as in (9.26) then there is no longer a conflict between efficiency and breakdown. As however the efficiency of the location functional increases the more it behaves like the mean with a corresponding increase in the bias function of (9.35) and (9.37). The conflict between efficiency and bias is a real one and gives rise to an optimality criterion, namely that of minimizing the bias subject to a lower bound on the efficiency. We refer to Martin and Zamar (1993a).

Outliers in \mathbb{R}

One of the main uses of robust functionals is the labelling of so called outliers (see Barnett and Lewis (1994), Hawkins (1980), Atkinson (1994), Gather (1990), Gather et al. (2003), and Simonoff (1984, 1987)). In the data of Table 9.1 the laboratories 1 and 3 are clearly outliers which should be flagged. The discussion in Sect. 9.1.1 already indicates that the mean and standard deviation are not appropriate tools for the identification of outliers as they themselves are so strongly influenced by the very outliers they are intended to identify. We now demonstrate this more precisely. One simple rule is to classify all observations more than three standard deviations from the mean as outliers. A simple calculation shows that this rule will fail to identify 10% arbitrarily large outliers with the same sign. More generally if all observations more than λ standard deviations from the mean are classified as outliers then this rule will fail to identify a proportion of $1/(1 + \lambda^2)$ outliers with the same sign. This is known as the masking effect (Pearson and Chandra Sekar (1936)) where the outliers mask their presence by distorting the mean and, more importantly, the standard deviation to such an extent as to render them useless for the detection of the outliers. One possibility is to choose a small value of λ but clearly if λ is too small then some non-outliers will be declared as outliers. In many cases the main body of the data can be well approximated by a normal distribution so we now investigate the choice of λ for samples of i.i.d. normal random variables. One possibility is to choose λ dependent on the sample size n so that with probability say 0.95 no observation will be flagged as an outlier. This leads to a value of λ of about $\sqrt{2 \log(n)}$ (Davies and Gather (1993)) and the largest proportion of one-sided outliers which can be detected is approximately $1/(1 + 2 \log(n))$ which tends to zero with n . It follows that there is no choice of λ which can detect say 10% outliers and at the same time not falsely flag non-outliers. In order to achieve this the mean and standard deviation must be replaced by functionals which are less effected by the outliers. In particular these functionals should be locally bounded (9.11). Considerations of asymptotic normality or efficiency are of little relevance here. Two obvious candidates are the median and MAD and if we use them instead of the mean and

standard deviation we are led to the identification rule (Hampel (1985)) of the form

$$|x_i - \text{MED}(\mathbf{x}_n)| \geq \lambda \text{MAD}(\mathbf{x}_n). \quad (9.60)$$

Hampel (1975) proposed setting $\lambda = 5.2$ as a general all purpose value. The concept of an outlier cannot in practice be very precise but in order to compare different identification rules we require a precise definition and a precise measure of performance. To do this we shall restrict attention to the normal model as one which is often reasonable for the main body of data. In other situations such as waiting times the exponential distribution may be more appropriate. The following is based on Davies and Gather (1993). To define an outlier we introduce the concept of an α -outlier. For the normal distribution $N(\mu, \sigma^2)$ and $\alpha \in (0, 1)$ we define the α -outlier region by

$$\text{out}(\alpha, N(\mu, \sigma^2)) = \{x \in \mathbb{R}: |x - \mu| > \sigma z_{1-\alpha/2}\}, \quad (9.61)$$

which is just the union of the lower and the upper $\alpha/2$ -tail regions. Here $z_{1-\alpha/2}$ denotes the $1 - \alpha/2$ -quantile of the standard normal distribution. For the exponential distribution $\text{Exp}(\lambda)$ with parameter λ we set

$$\text{out}(\alpha, \text{Exp}(\lambda)) = \{x \in \mathbb{R}: x > -\lambda \ln \alpha\} \quad (9.62)$$

which is the upper α -tail region (Gather and Schultze (1999)). The extension to other distributions P is clear. Each point located in the outlier region is called an α -outlier, otherwise it is called an α -inlier. This definition of an outlier refers only to its position in relation to the statistical model for the good data. No assumptions are made concerning the distribution of these outliers or the mechanism by which they are generated.

We can now formulate the task of outlier identification for the normal distribution as follows: For a given sample $\mathbf{x}_n = (x_1, \dots, x_n)$ which contains at least $[n/2] + 1$ i.i.d. observations distributed according to $N(\mu, \sigma^2)$, we have to find all those x_i that are located in $\text{out}(\alpha, N(\mu, \sigma^2))$. The level α can be chosen to be dependent on the sample size. If for some $\tilde{\alpha} \in (0, 1)$ we set

$$\alpha = \alpha_n = 1 - (1 - \tilde{\alpha})^{1/n}, \quad (9.63)$$

then the probability of finding at least one observation of a $N(\mu, \sigma^2)$ -sample of size n within $\text{out}(\alpha_n, N(\mu, \sigma^2))$ is not larger than $\tilde{\alpha}$. Consider now the general Hampel identifier which classifies all observations x_i in

$$\text{OR}^H(\mathbf{x}_n, \alpha_n) = \{x \in \mathbb{R}: |x - \text{Med}(\mathbf{x}_n)| > g_n(\alpha_n) \text{MAD}(\mathbf{x}_n)\} \quad (9.64)$$

as outliers. The region $\text{OR}^H(\mathbf{x}_n, \alpha_n)$ may be regarded as an empirical version of the outlier region $\text{out}(\alpha_n, N(\mu, \sigma^2))$. The constant $g_n(\alpha_n)$ standardizes the behaviour of the procedure for i.i.d. normal samples which may be done in several ways. One

is to determine the constant so that with probability at least $1 - \bar{\alpha}$ no observation X_i is identified as an outlier, that is

$$P(X_i \notin \text{OR}(X_n, \alpha_n), i = 1, \dots, n) \geq 1 - \bar{\alpha}. \quad (9.65)$$

A second possibility is to require that

$$P(\text{OR}(X_n, \alpha_n) \subset \text{out}(\alpha_n, P)) \geq 1 - \bar{\alpha}. \quad (9.66)$$

If we use (9.65) and set $\bar{\alpha} = 0.05$ then for $n = 20, 50$ and 100 simulations give $g_n(\alpha_n) = 5.82, 5.53$ and 5.52 respectively. For $n > 10$ the normalizing constants $g_n(\alpha_n)$ can also be approximated according to the equations given in Sect. 5 of Gather (1990).

To describe the worst case behaviour of an outlier identifier we can look at the largest nonidentifiable outlier, which it allows. From Davies and Gather (1993) we report some values of this quantity for the Hampel identifier (HAMP) and contrast them with the corresponding values of a sophisticated high breakdown point outwards testing identifier (ROS), based on the non-robust mean and standard deviation (Rosner (1975); Tietjen and Moore (1972)). Both identifiers are standardized by (9.65) with $\bar{\alpha} = 0.05$. Outliers are then observations with absolute values greater than $3.016(n = 20), 3.284(n = 50)$ and $3.474(n = 100)$. For $k = 2$ outliers and $n = 20$ the average sizes of the largest non-detected outlier are 6.68 (HAMP) and 8.77 (ROS), for $k = 5$ outliers and $n = 50$ the corresponding values are 4.64 (HAMP) and 5.91 (ROS) and finally for $k = 15$ outliers and $n = 100$ the values are 5.07 (HAMP) and 9.29 (ROS).

Location and Scale in \mathbb{R}^k

9.3

Equivariance and Metrics

9.3.1

In Sect. 9.2.1 we discussed the equivariance of estimators for location and scale with respect to the affine group of transformations on \mathbb{R} . This carries over to higher dimensions although here the requirement of affine equivariance lacks immediate plausibility. A change of location and scale for each individual component in \mathbb{R}^k is represented by an affine transformation of the form $A(x) + b$ where A is a diagonal matrix. A general affine transformation forms linear combinations of the individual components which goes beyond arguments based on units of measurement. The use of affine equivariance reduces to the almost empirical question as to whether the data, regarded as a cloud of points in \mathbb{R}^k , can be well represented by an ellipsoid. If this is the case as it often is then consideration of linear combinations of different components makes data analytical sense. With this proviso in mind we consider the affine group \mathcal{A} of transformations of \mathbb{R}^k into itself,

$$\mathcal{A} = \{A : \mathcal{A}(x) = A(x) + b\}, \quad (9.67)$$

where A is a non-singular $k \times k$ -matrix and b is an arbitrary point in \mathbb{R}^k . Let \mathcal{P}'_k denote a family of distributions over \mathbb{R}^k which is closed under affine transformations

$$P \in \mathcal{P}'_k \Rightarrow P^{\mathcal{A}} \in \mathcal{P}'_k, \text{ for all } \mathcal{A} \in \mathcal{A}. \quad (9.68)$$

A function $T_l : \mathcal{P}'_k \rightarrow \mathbb{R}^k$ is called a location functional if it is well defined and

$$T_l(P^{\mathcal{A}}) = \mathcal{A}(T_l(P)), \text{ for all } \mathcal{A} \in \mathcal{A}, P \in \mathcal{P}'_k. \quad (9.69)$$

A functional $T_s : \mathcal{P}'_k \rightarrow \Sigma_k$ where Σ_k denotes the set of all strictly positive definite symmetric $k \times k$ matrices is called a scale or scatter functional if

$$T_s(P^{\mathcal{A}}) = \mathcal{A}T_s(P)\mathcal{A}^{\top}, \text{ for all } \mathcal{A} \in \mathcal{A}, P \in \mathcal{P}'_k \text{ with } \mathcal{A}(x) = A(x) + b. \quad (9.70)$$

The requirement of affine equivariance is a strong one as we now indicate. The most obvious way of defining the median of a k -dimensional data set is to define it by the medians of the individual components. With this definition the median is equivariant with respect to transformations of the form $\Lambda(x) + b$ with Λ a diagonal matrix but it is not equivariant for the affine group. A second possibility is to define the median of a distribution P by

$$\text{MED}(P) = \underset{\mu}{\text{argmin}} \int (\|x - \mu\| - \|x\|) dP(x).$$

With this definition the median is equivariant with respect to transformations of the form $x \rightarrow O(x) + b$ with O an orthogonal matrix but not with respect to the affine group or the group $x \rightarrow \Lambda(x) + b$ with Λ a diagonal matrix. The conclusion is that there is no canonical extension of the median to higher dimensions which is equivariant with respect to the affine group.

In Sect. 9.2 use was made of metrics on the space of probability distributions on \mathbb{R} . We extend this to \mathbb{R}^k where all metrics we consider are of the form

$$d_{\mathcal{C}}(P, Q) = \sup_{C \in \mathcal{C}} |P(C) - Q(C)| \quad (9.71)$$

where \mathcal{C} is a so called Vapnik–Cervonenkis class (see for example Pollard (1984)). The class \mathcal{C} can be chosen to suit the problem. Examples are the class of all lower dimensional hyperplanes

$$\mathcal{H} = \{H : H \text{ lower dimensional hyperplane}\} \quad (9.72)$$

and the class of all ellipsoids

$$\mathcal{E} = \{E : E \text{ an ellipsoid}\}. \quad (9.73)$$

These give rise to the metrics $d_{\mathcal{H}}$ and $d_{\mathcal{E}}$ respectively. Just as in \mathbb{R} metrics $d_{\mathcal{C}}$ of the form (9.71) allow direct comparisons between empirical measures and models. We have

$$d_{\mathcal{C}}(P_n(P), P) = O(1/\sqrt{n}) \quad (9.74)$$

uniformly in P (see Pollard (1984)).

M-estimators of Location and Scale

Given the usefulness of M-estimators for one dimensional data it seems natural to extend the concept to higher dimensions. We follow Maronna (1976). For any positive definite symmetric $k \times k$ -matrix Σ we define the metric $d(\cdot, \cdot : \Sigma)$ by

$$d(x, y : \Sigma)^2 = (x - y)^\top \Sigma^{-1} (x - y), \quad x, y \in \mathbb{R}^k.$$

Further, let u_1 and u_2 be two non-negative continuous functions defined on \mathbb{R}_+ and be such that $su_i(s), s \in \mathbb{R}_+, i = 1, 2$ are both bounded. For a given probability distribution P on the Borel sets of \mathbb{R}^k we consider in analogy to (9.21) and (9.22) the two equations in μ and Σ

$$\int (x - \mu) u_1(d(x, \mu; \Sigma)) dP = 0. \quad (9.75)$$

$$\int u_2(d(x, \mu; \Sigma)^2) (x - \mu)(x - \mu)^\top dP = 0. \quad (9.76)$$

Assuming that at least one solution (μ, Σ) exists we denote it by $T_M(P) = (\mu, \Sigma)$. The existence of a solution of (9.75) and (9.76) can be shown under weak conditions as follows. If we define

$$\Delta(P) = \max\{P(H) : H \in \mathcal{H}\} \quad (9.77)$$

with \mathcal{H} as in (9.73) then a solution exists if $\Delta(P) < 1 - \delta$ where δ depends only on the functions u_1 and u_2 (Maronna (1976)). Unfortunately the problem of uniqueness is much more difficult than in the one-dimensional case. The conditions placed on P in Maronna (1976) are either that it has a density $f_P(x)$ which is a decreasing function of $\|x\|$ or that it is symmetric $P(B) = P(-B)$ for every Borel set B . Such conditions do not hold for real data sets which puts us in an awkward position. Furthermore without existence and uniqueness there can be no results on asymptotic normality and consequently no results on confidence intervals. The situation is unsatisfactory so we now turn to the one class of M-functionals for which existence and uniqueness can be shown. The following is based on Kent and Tyler (1991) and is the multidimensional generalization of (9.33). The k -dimensional t -distribution with density $f_{k,v}(\cdot : \mu, \Sigma)$ is defined by

$$f_{k,v}(x : \mu, \Sigma) = \frac{\Gamma(\frac{1}{2}(v+k))}{(vk)^{k/2} \Gamma(\frac{1}{2}v)} |\Sigma|^{-\frac{1}{2}} \left(1 + \frac{1}{v} (x - \mu)^\top \Sigma^{-1} (x - \mu) \right)^{-\frac{1}{2}(v+k)} \quad (9.78)$$

and we consider the minimization problem

$$T_M(p) = (T_l(P), T_s(P)) = \operatorname{argmin}_{\mu, \Sigma} \int f_{k,v}(x : \mu, \Sigma) dP(x) + \frac{1}{2} \log(|\Sigma|) \quad (9.79)$$

where $|\Sigma|$ denotes the determinant of the positive definite matrix Σ . For any distribution P on the Borel sets of \mathbb{R}^k we define $\Delta(P)$ which is the k -dimensional version of (9.23). It can be shown that if $\Delta(P) < 1/2$ then (9.79) has a unique solution. More-

over for data sets there is a simple algorithm which converges to the solution. On differentiating the right hand side of (9.79) it is seen that the solution is an M-estimator as in (9.75) and (9.76). Although this has not been proven explicitly it seems clear that the solution will be locally uniformly Fréchet differentiable, that is, it will satisfy (9.12) where the influence function $I(x, T_M, P)$ can be obtained as in (9.54) and the metric d_{k_0} is replaced by the metric $d_{\mathcal{H}}$. This together with (9.74) leads to uniform asymptotic normality and allows the construction of confidence regions. The only weakness of the proposal is the low gross error breakdown point $\varepsilon^*(T_M, P, GE)$ defined below which is at most $1/(k+1)$. This upper bound is shared with the M-functionals defined by (9.75) and (9.76) (Maronna (1976)). The problem of constructing high breakdown functionals in k dimensions will be discussed below.

9.3.3 Bias and Breakdown

The concepts of bias and breakdown developed in Sect. 9.2.4 carry over to higher dimensions. Given a metric d on the space of distributions on \mathbb{R}^k and a location functional T_l we follow (9.37) and define

$$b(T_l, P, \varepsilon, d) = \sup\{\|T_l(Q)\| : d(P, Q) < \varepsilon\} \quad (9.80)$$

and

$$b(T_l, P, \varepsilon, GE) = \sup\{\|T_l(Q)\| : Q = (1 - \varepsilon)P + \varepsilon G, G \in \mathcal{P}\}, \quad (9.81)$$

where by convention $\|T_l(Q)\| = \infty$ if T_l is not defined at Q . The extension to scale functionals is not so obvious as there is no canonical definition of bias. We require a measure of difference between two positive definite symmetric matrices. For reasons of simplicity and because it is sufficient for our purposes the one we take is $|\log(|\Sigma_1|/|\Sigma_2|)|$. Corresponding to (9.36) we define

$$b(T_s, P, \varepsilon, d) = \sup\{|\log(|T_s(Q)|/|T_s(P)|)| : d(P, Q) < \varepsilon\} \quad (9.82)$$

and

$$b(T_s, P, \varepsilon, GE) = \sup\{|\log(|T_s(Q)|/|T_s(P)|)| : Q = (1 - \varepsilon)P + \varepsilon G, G \in \mathcal{P}\}. \quad (9.83)$$

Most work is done using the gross error model (9.81) and (9.83). The breakdown points of T_l are defined by

$$\varepsilon^*(T_l, P, d) = \sup\{\varepsilon : b(T_l, P, \varepsilon, d) < \infty\} \quad (9.84)$$

$$\varepsilon^*(T_l, P, GE) = \sup\{\varepsilon : b(T_l, P, \varepsilon, GE) < \infty\} \quad (9.85)$$

$$\varepsilon^*(T_l, P_n, fsbp) = \max\{k/n : |T_l(P_n^k)| < \infty\}, \quad (9.86)$$

where (9.86) corresponds in the obvious manner to (9.41). The breakdown points for the scale functional T_s are defined analogously using the bias functional (9.82). We have

Theorem 4 For any translation equivariant functional T_l

4

$$\varepsilon^*(T_l, P, d_{\mathcal{H}}) \leq 1/2 \text{ and } \varepsilon^*(T_l, P_n, f_{sbp}) \leq \lfloor n/2 \rfloor / n \quad (9.87)$$

and for any affine equivariant scale functional

$$\varepsilon^*(T_s, P, d_{\mathcal{E}}) \leq (1 - \Delta(P))/2 \text{ and } \varepsilon^*(T_s, P_n, f_{sbp}) \leq (1 - \Delta(P_n))/2. \quad (9.88)$$

In Sect. 9.2.4 it was shown that the M-estimators of Sect. 9.2.3 can attain or almost attain the upper bounds of Theorem 1. Unfortunately this is not the case in k dimensions where as we have already mentioned the breakdown points of M-functionals of Sect. 9.3.2 are at most $1/(k+1)$. In recent years much research activity has been directed towards finding high breakdown affinely equivariant location and scale functionals which attain or nearly attain the upper bounds of Theorem 4. This is discussed in the next section.

High Breakdown Location and Scale Functionals in \mathbb{R}^k

9.3.4

The first high breakdown affine equivariant location and scale functionals were proposed independently of each other by Stahel (1981) and Donoho (1982). They were defined for empirical data but the construction can be carried over to measures satisfying a certain weak condition. The idea is to project the data points onto lines through the origin and then to determine which points are outliers with respect to this projection using one-dimensional functions with a high breakdown point. More precisely we set

$$o(x_i, \theta) = |x_i^\top \theta - \text{MED}(x_1^\top \theta, \dots, x_n^\top \theta)| / \text{MAD}(x_1^\top \theta, \dots, x_n^\top \theta) \quad (9.89)$$

and

$$o(x_i) = \sup\{o(x_i, \theta) : \|\theta\| = 1\}. \quad (9.90)$$

This is a measure for the outlyingness of the point x_i and it may be checked that it is affine invariant. Location and scale functionals may now be obtained by taking for example the mean and the covariance matrix of those $\lfloor n/2 + 1 \rfloor$ observations with the smallest outlyingness measure. Although (9.90) requires a supremum over all values of θ this can be reduced for empirical distributions as follows. Choose all linearly independent subsets x_{i_1}, \dots, x_{i_k} of size k and for each such subset determine a θ which is orthogonal to their span. If the sup in (9.90) is replaced by a maximum over all such θ then the location and scale functionals remain affine equivariant and retain the high breakdown point. Although this requires the consideration of only a finite number of directions namely at most $\binom{n}{k}$ this number is too large to make it a practicable possibility even for small values of n and k . The problem of calculability has remained with high breakdown methods ever since and it is their main weakness. There are still no high breakdown affine equivariant functionals

which can be calculated exactly except for very small data sets. Huber (1995) goes as far as to say that the problem of calculability is the breakdown of high breakdown methods. This is perhaps too pessimistic but the problem remains unsolved.

Rousseeuw (1985) introduced two further high breakdown location and scale functionals as follows. The first, the so called minimum volume ellipsoid (MVE) functional, is a multidimensional version of Tukey's shortest half-sample (9.8) and is defined as follows. We set

$$E = \operatorname{argmin}_{\tilde{E}} \{ |\tilde{E}| : |\{i : x_i \in \tilde{E}\}| \geq \lfloor n/2 \rfloor \}, \quad (9.91)$$

where $|E|$ denotes the volume of E and $|\{ \cdot \}|$ denotes the number of elements of the set $\{ \cdot \}$. In other words E has the smallest volume of any ellipsoid which contains more than half the data points. For a general distribution P we define

$$E(P) = \operatorname{argmin}_{\tilde{E}} \left\{ |\tilde{E}| : \int_{\tilde{E}} dP \geq 1/2 \right\}. \quad (9.92)$$

Given E the location functional $T_l(P)$ is defined to be the centre $\mu(E)$ of E and the covariance functional $T_s(P)$ is taken to be $c(k)\Sigma(E)$ where

$$E = \{x : (x - \mu(E))^\top \Sigma^{-1}(x - \mu(E)) \leq 1\}. \quad (9.93)$$

The factor $c(k)$ can be chosen so that $c(k)\Sigma(E) = I_k$ for the standard normal distribution in k dimensions.

The second functional is based on the so called minimum covariance determinant (MCD) and is as follows. We write

$$\mu(B) = \int_B x dP(x)/P(B) \quad (9.94)$$

$$\Sigma(B) = \int_B (x - \mu(B))(x - \mu(B))^\top dP(x)/P(B) \quad (9.95)$$

and define

$$\operatorname{MCD}(P) = \operatorname{argmin}_B \{ |\Sigma(B)| : P(B) \geq 1/2 \}, \quad (9.96)$$

where $|\Sigma(B)|$ is defined to be infinite if either of (9.94) or (9.95) does not exist. The location functional is taken to be $\mu(\operatorname{MCD}(B))$ and the scatter functional $c(k)\Sigma(\operatorname{MCD}(B))$ where again $c(k)$ is usually chosen so that $c(k)\Sigma(\operatorname{MCD}(B)) = I_k$ for the standard normal distribution in k -dimensions. It can be shown that both these functionals are affinely equivariant.

A smoothed version of the minimum volume estimator can be obtained by considering the minimization problem

$$\text{minimize } |\Sigma| \text{ subject to } \int \varrho((x - \mu)^\top \Sigma^{-1}(x - \mu)) dP(x) \geq 1/2, \quad (9.97)$$

where $\varphi : \mathbb{R}_+ \rightarrow [0, 1]$ satisfies $\varphi(0) = 1$, $\lim_{x \rightarrow \infty} \varphi(x) = 0$ and is continuous on the right (see Davies (1987)). This gives rise to the class of so called S -functionals. The minimum volume estimator can be obtained by specializing to the case $\varphi(x) = \{0 \leq x < 1\}$.

On differentiating (9.97) it can be seen that an S -functional can be regarded as an M -functional but with redescending functions u_1 and u_2 in contrast to the conditions placed on u_1 and u_2 in (9.75) and (9.76) (Lopuhaä (1989)). For such functions the defining equations for an M -estimator have many solutions and the minimization problem of (9.97) can be viewed as a choice function. Other choice functions can be made giving rise to different high breakdown M -estimators. We refer to Lopuhaä (1991) and Kent and Tyler (1996). A further class of location and scatter functionals have been developed from Tukey's concept of depth (Tukey (1975)). We refer to Donoho and Gasko (1992), Liu et al. (1999) and Zuo and Serfling (2000a, 2000b). Many of the above functionals have breakdown points close to or equal to the upper bound of Theorem 4. For the calculation of breakdown points we refer to Davies (1987, 1993), Lopuhaä and Rousseeuw (1991), Donoho and Gasko (1992) and Tyler (1994).

The problem of determining a functional which minimizes the bias over a neighbourhood was considered in the one-dimensional case in Sect. 9.2.4. The problem is much more difficult in \mathbb{R}^k but some work in this direction has been done (see Adrover (1998)). The more tractable problem of determining the size of the bias function for particular functionals or classes of functionals has also been considered (Yohai and Maronna (1990); Maronna et al. (1992)).

All the above functionals can be shown to exist but there are problems concerning the uniqueness of the functional. Just as in the case of Tukey's shortest half (9.8) restrictions must be placed on the distribution P which generally include the existence of a density with given properties (see Davies (1987) and Tatsuoka and Tyler (2000)) and which is therefore at odds with the spirit of robust statistics. Moreover even uniqueness and asymptotic normality at some small class of models are not sufficient. Ideally the functional should exist and be uniquely defined and locally uniformly Fréchet differentiable just as in Sect. 9.2.5. It is not easy to construct affinely equivariant location and scatter functionals which satisfy the first two conditions but it has been accomplished by Dietel (1993) using the Stahel–Donoho idea of projections described above. To go further and define functionals which are also locally uniformly Fréchet differentiable with respect to some metric d_c just as in the one-dimensional case considered in Sect. 9.2.5 is a very difficult problem. The only result in this direction is again due to Dietel (1993) who managed to construct functionals which are locally uniformly Lipschitz. The lack of locally uniform Fréchet differentiability means that all derived confidence intervals will exhibit a certain degree of instability. Moreover the problem is compounded by the inability to calculate the functionals themselves. To some extent it is possible to reduce the instability by say using the MCD functional in preference to the MVE functional, by reweighting the observations or by calculating a one-step M -functional as in (9.29) (see Davies (1992a)). However the problem remains and for this reason we do not discuss the research which has been carried out on the

efficiency of the location and scatter functionals mentioned above. Their main use is in data analysis where they are an invaluable tool for detecting outliers. This will be discussed in the following section.

A scatter matrix plays an important role in many statistical techniques such as principal component analysis and factor analysis. The use of robust scatter functionals in some of these areas has been studied by among others Croux and Haesbroeck (2000), Croux and Dehon (2001) and Willems et al. (2002).

As already mentioned the major weakness of all known high breakdown functionals is their computational complexity. For the MCD functional an exact algorithm of the order of $n^{k(k+3)/2}$ exists and there are reasons for supposing that this cannot be reduced to below n^k (Bernholt and Fischer (2001)). This means that in practice for all but very small data sets heuristic algorithms have to be used. We refer to Rousseeuw and Van Driessen (1999) for a heuristic algorithm for the MCD-functional.

9.3.5 Outliers in \mathbb{R}

Whereas for univariate, bivariate and even trivariate data outliers may often be found by visual inspection, this is not practical in higher dimensions (Caroni and Prescott (1992); Hadi (1994); Barme-Delcroix and Gather (2000); Gnanadesikan and Kettenring (1972); Hadi and Simonoff (1997)). This makes it all the more important to have methods which automatically detect high dimensional outliers. Much of the analysis of the one-dimensional problem given in Sect. 9.2.7 carries over to the k -dimensional problem. In particular outlier identification rules based on the mean and covariance of the data suffer from masking problems and must be replaced by high breakdown functionals (see also Rocke and Woodruff (1996, 1997)). We restrict attention to affine equivariant functionals so that an affine transformation of the data will not alter the observations which are identified as outliers. The identification rules we consider are of the form

$$(x_i - T_l(P_n))^T T_s(P_n)^{-1} (x_i - T_l(P_n)) \geq c(k, n), \quad (9.98)$$

where P_n is the empirical measure, T_l and T_s are affine equivariant location and scatter functionals respectively and $c(k, n)$ is a constant to be determined. This rule is the k -dimensional counterpart of (9.60). In order to specify some reasonable value for $c(k, n)$ and in order to be able to compare different outlier identifiers we require, just as in Sect. 9.2.7, a precise definition of an outlier and a basic model for the majority of the observations. As our basic model we take the k -dimensional normal distribution $\mathcal{N}(\mu, \Sigma)$. The definition of an α_n -outlier corresponds to (9.62) and is

$$\text{out}(\alpha_n, \mu, \Sigma) = \{x \in \mathbb{R}^k : (x - \mu)^T \Sigma^{-1} (x - \mu) > \chi_{k; 1-\alpha_n}^2\}, \quad (9.99)$$

where $\alpha_n = 1 - (1 - \tilde{\alpha})^{1/n}$ for some given value of $\tilde{\alpha} \in (0, 1)$. Clearly for an i.i.d. sample of size n distributed according to $\mathcal{N}(\mu, \Sigma)$ the probability that no

observation lies in the outlier region of (9.99) is just $1 - \alpha$. Given location and scale functionals T_l and T_s and a sample \tilde{x}_n we write

$$\text{OR}^H(\tilde{x}_n, \alpha_n) = \{x \in \mathbb{R}^k : (x - T_l(P_n))^\top T_s(P_n)^{-1} (x - T_l(P_n)) \geq c(k, n, \alpha_n)\} \quad (9.100)$$

which corresponds to (9.64). The region $\text{OR}^H(\tilde{x}_n, \alpha_n)$ is the empirical counterpart of $\text{out}(\alpha_n, \mu, \Sigma)$ of (9.99) and any observation lying in $\text{OR}^H(\tilde{x}_n, \alpha_n)$ will be identified as an outlier. Just as in the one-dimensional case we determine the $c(k, n, \alpha_n)$ by requiring that with probability $1 - \tilde{\alpha}$ no observation is identified as an outlier in i.i.d. $\mathcal{N}(\mu, \Sigma)$ samples of size n . This can be done by simulations with appropriate asymptotic approximations for large n . The simulations will of course be based on the algorithms used to calculate the functionals and will not be based on the exact functionals assuming these to be well defined. For the purpose of outlier identification this will not be of great consequence. We give results for three multivariate outlier identifiers based on the MVE- and MCD-functionals of Rousseeuw (1985) and the S -functional based on Tukey's biweight function as given in Rocke (1996). There are good heuristic algorithms for calculating these functionals at least approximately (Rocke (1996); Rousseeuw and Van Driessen (1999); Rousseeuw and van Zoomeeren (1990)). The following is based on Becker and Gather (2001). Table 9.2 gives the values of $c(k, n, \alpha_n)$ with $\alpha = 0.1$. The results are based on 10,000 simulations for each combination of k and n .

Becker and Gather (2001) show by simulations that although none of the above rules fails to detect arbitrarily large outliers it still can happen that very extreme observations are not identified as outliers. To quantify this we consider the identifier OR_{MVE} and the constellation $n = 50, k = 2$ with $m = 5$ observations replaced by other values. The mean norm of the most extreme nonidentifiable outlier is 4.17. The situation clearly becomes worse with an increasing proportion of replaced observations and with the dimension k (see Becker and Gather (1999)). If we use the mean of the norm of the most extreme non-identifiable outlier as a criterion then none of the three rules dominates the others although the biweight identifier performs reasonably well in all cases and is our preferred choice.

Table 9.2. Normalizing constants $c(k, n, \alpha_n)$ for OR_{MVE} , OR_{MCD} , OR_{BW} for $\alpha = 0.1$

n	k	c_{MVE}	c_{MCD}	c_{BW}
20	2	19.14222	85.58786	21.35944
20	3	23.47072	167.61310	26.87044
20	4	33.72110	388.84680	33.17018
50	2	17.54896	28.51695	16.93195
50	3	20.61580	41.83594	19.78682
50	4	24.65417	64.18462	23.14061

9.4 Linear Regression

9.4.1 Equivariance and Metrics

The linear regression model may be written in the form

$$Y_i = \mathbf{x}_i^\top \boldsymbol{\beta} + \varepsilon_i, \quad i = 1, \dots, n \quad (9.101)$$

where \mathbf{x}_i , $i = 1 \dots, n$ and $\boldsymbol{\beta} \in \mathbb{R}^k$. The assumptions of the standard model are that the x_i are fixed and that the ε_i are i.i.d. random variables with the default distribution being the normal distribution $N(0, \sigma^2)$. There are of course many other models in the literature including random x_i -values and a covariance structure for the errors ε_i . For the purpose of robust regression we consider probability distributions P on \mathbb{R}^{k+1} where the first k components refer to the covariates \mathbf{x} and the last component is the corresponding value of y . We restrict attention to the family \mathcal{P}_{k+1} of probability measures given by

$$\mathcal{P}_{k+1} = \{P : P(H \times \mathbb{R}) < 1 \text{ for all lower dimensional subspaces } H \subset \mathbb{R}^k\}. \quad (9.102)$$

The metric we use on \mathcal{P}_{k+1} is $d_{\mathcal{J}\ell}$ with \mathcal{H} given by (9.73).

Consider the regression group G of transformations $g : \mathbb{R}^{k+1} \rightarrow \mathbb{R}^{k+1}$ of the form

$$g(\mathbf{x}, y) = (A(\mathbf{x}), s y + \mathbf{x}^\top \boldsymbol{\gamma}) \quad (9.103)$$

where A is a non-singular $k \times k$ -matrix, $s \in \mathbb{R}$, $s \neq 0$, and $\boldsymbol{\gamma} \in \mathbb{R}^k$. A functional $T : \mathcal{P}_{k+1} \rightarrow \mathbb{R}^k \times \mathbb{R}_+$ is called a regression functional if for all $g \in G$ and $P \in \mathcal{P}_{k+1}$

$$T(P^g) = h_g(T(P)), \quad (9.104)$$

where

$$h_g(\boldsymbol{\beta}, \sigma) = (s(A^{-1})^\top(\boldsymbol{\beta} + \boldsymbol{\gamma}), s\sigma). \quad (9.105)$$

with A and $\boldsymbol{\gamma}$ as in (9.103). The first k components of $T(P)$ specify the value of $\boldsymbol{\beta} \in \mathbb{R}^k$ and the last component that of σ . The restriction to models $P \in \mathcal{P}_{k+1}$ of (9.102) is that without such a restriction there is no uniquely defined value of $\boldsymbol{\beta}$.

9.4.2 M-estimators for Regression

Given a distribution $P \in \mathcal{P}_{k+1}$ we define an M-functional by $T(P) = (\boldsymbol{\beta}^*, \sigma^*)$ where $(\boldsymbol{\beta}^*, \sigma^*)$ is a solution of the equations

$$\int \phi(\mathbf{x}, (y - \mathbf{x}^\top \boldsymbol{\beta}) / \sigma) \mathbf{x} dP(\mathbf{x}, y) = 0, \quad (9.106)$$

$$\int \chi((y - \mathbf{x}^\top \boldsymbol{\beta}) / \sigma) dP(\mathbf{x}, y) = 0 \quad (9.107)$$

for given functions $\phi : \mathbb{R}^{k+1} \rightarrow \mathbb{R}$ and $\chi : \mathbb{R} \rightarrow \mathbb{R}$. Just as in Sect. 9.3.2 for M-functionals of location and scatter there are problems concerning the existence and uniqueness. Maronna and Yohai (1981) give sufficient conditions for existence which depend only on the properties of ϕ and χ and the values of $\sup_{\theta} \{P(\theta^\top \mathbf{x} = 0) : \theta \neq 0\}$ and $\sup_{\alpha, \theta} \{P(\alpha y + \theta^\top \mathbf{x} = 0) : |\alpha| + \|\theta\| \neq 0\}$. Uniqueness requires additional strong assumptions such as either symmetry or the existence of a density for the conditional distribution of $y - \theta_0^\top \mathbf{x}$ for each fixed \mathbf{x} . Huber (1981) considers the minimization problem

$$(\beta^*, \sigma^*) = \operatorname{argmin} \left(\int \varphi((y - \mathbf{x}^\top \beta) / \sigma) dP(\mathbf{x}, y) + a \right) \sigma, \tag{9.108}$$

where $\varphi : \mathbb{R} \rightarrow \mathbb{R}_+$ is convex with $\varphi(0) = 0$ and $a > 0$. Under appropriate conditions on φ it can be shown that the solution is unique and that there exists a convergent algorithm to calculate it. On differentiating (9.108) we obtain (9.106) and (9.107) with

$$\phi(\mathbf{x}, u) = \varphi^{(1)}(u) \text{ and } \chi(u) = u\varphi^{(1)}(u) - \varphi(u) - a. \tag{9.109}$$

Even if the solution of (9.106) and (9.107) exists and is unique it is not necessarily regression equivariant. To make it so we must introduce a scatter functional T_Σ on the marginal distributions $P', P'(B) = P(B \times \mathbb{R})$ of the covariate \mathbf{x} . Such a functional satisfies $T_\Sigma(P'^A) = AT_\Sigma(P')A^\top$ for any non-singular $k \times k$ -matrix A and is required not only for equivariance reasons but also to downweight outlying \mathbf{x} -values or so called leverage points. For this latter purpose the functional T_Σ must also be robust. We now replace (9.106) by

$$\int \phi(\mathbf{x}^\top T_\Sigma(P)^{-1} \mathbf{x}, (y - \mathbf{x}^\top \beta) / \sigma) \mathbf{x} dP(\mathbf{x}, y) = 0. \tag{9.110}$$

The resulting functional is now regression equivariant but its analysis is more difficult requiring as it does an analysis of the robustness properties of the scatter functional T_Σ .

Finally we note that in the literature most ϕ functions of (9.106) are of the form

$$\phi(\mathbf{x}, u) = \pi(\mathbf{x})\psi(u) \tag{9.111}$$

and the resulting functionals are known as GM-functionals. We refer to Hampel et al. (1986).

Bias and Breakdown

9.4.3

Given a regression functional $T_r = (T_b, T_s)$ where T_b refers to the β -components and T_s is the scale part it is usual to define breakdown just by the behaviour of T_b and to neglect T_s . The breakdown point of T_r at the distribution P is defined by

$$\varepsilon^*(T_r, P, d_{\mathcal{H}}) = \sup\{\varepsilon : b(T_r, P, \varepsilon, d_{\mathcal{H}}) < \infty\} \tag{9.112}$$

where

$$b(T_r, P, \varepsilon, d_{\mathcal{H}}) = \sup\{\|T_b(Q) - T_b(P)\| : d_{\mathcal{H}}(P, Q) < \varepsilon\} \quad (9.113)$$

with corresponding definitions for the gross error neighbourhood $\varepsilon^*(T_r, P, GE)$ and for the finite sample breakdown point $\varepsilon^*(T_r, P_n, fsbp)$. To state the next theorem we set

$$\Delta(P) = \sup\{P(H \times \mathbb{R}) : H \text{ a plane in } \mathbb{R}^k \text{ of dimension at most } k-1\},$$

which is the regression equivalent of (9.77). We have

5 Theorem 5 For any regression equivariant functional

$$\varepsilon^*(T_r, P, d_h) \leq (1 - \Delta(P))/2 \text{ and } \varepsilon^*(T_r, P_n, fsbp) \leq (1 - \Delta(P_n))/2. \quad (9.114)$$

If one considers L_1 -regression

$$\beta^* = \operatorname{argmin} \sum_{i=1}^n |y_i - \mathbf{x}_i^\top \beta| \quad (9.115)$$

it can be shown if one \mathbf{x}_i is sufficiently outlying then the residual at this point will be zero and hence the finite sample breakdown point is a disappointing $1/n$. This turns out to apply to most M-functionals of the last section whose breakdown point is at most $1/(k+1)$ irrespective of their exact definition. The literature on this point is unsatisfactory. Although some M-functionals have been shown to have a positive breakdown point this has only been done under the assumption that the scale part T_s is known. As obtaining the correct magnitude of the scale of the errors is in some sense the most difficult problem in robust regression such results are of limited value. They do not however alter the fact that M-functionals have a disappointing breakdown point. We now turn to the problem of constructing high breakdown regression functionals.

9.4.4 High Breakdown Regression Functionals

The first high breakdown regression functional was proposed by Hampel (1975) and is as follows.

$$T_{lms}(P) = \operatorname{argmin}_{(\beta, \sigma)} \left\{ \sigma : \int \{|y - \mathbf{x}^\top \beta| \leq \sigma\} dP(\mathbf{x}, y) \geq 1/2 \right\}. \quad (9.116)$$

The idea goes back to Tukey's shortest half-sample of which it is the regression counterpart. It can be shown that it has almost the highest finite sample breakdown point given by Theorem 5. By slightly altering the factor $1/2$ in (9.116) to take into account the dimension k of the \mathbf{x} -variables it can attain this bound. Rousseeuw (1984) propagated its use and gave it the name by which it is now known, the least median of squares LMS. Rousseeuw calculated the finite sample breakdown point

and provided a first heuristic algorithm which could be applied to real data sets. He also defined a second high breakdown functional known as least trimmed squares LTS defined by

$$T_{lts}(P) = \operatorname{argmin}_{(\beta, \sigma)} \left\{ \int (y - \mathbf{x}^\top \beta)^2 \{ |y - \mathbf{x}^\top \beta| \leq \sigma \} dP(\mathbf{x}, y) : \int \{ |y - \mathbf{x}^\top \beta| \leq \sigma \} dP(\mathbf{x}, y) \geq 1/2 \right\}. \quad (9.117)$$

There are now many high breakdown regression functionals such as S -functionals (Rousseeuw and Yohai (1984)), MM-functionals (Yohai (1987)), τ -functionals (Yohai and Zamar (1988)), constrained M-functionals (Mendes and Tyler (1996)), rank regression (Chang et al. (1999)) and regression depth (Rousseeuw and Hubert (1999)). Just as in the location and scale problem in \mathbb{R}^k statistical functionals can have the same breakdown points but very different bias functions. We refer to Martin et al. (1989), Maronna and Yohai (1993) and Berrendero and Zamar (2001). All these high breakdown functionals either attain or by some minor adjustment can be made to attain the breakdown points of Theorem 5 with the exception of depth based methods where the maximal breakdown point is $1/3$ (see Donoho and Gasko (1992)).

All the above high breakdown regression functionals can be shown to exist under weak assumptions but just as in the case of high breakdown location and scatter functionals in \mathbb{R}^k uniqueness can only be shown under very strong conditions which typically involve the existence of a density function for the errors (see Davies (1993)). The comments made about high breakdown location and scale functionals in \mathbb{R}^k apply here. Thus even if a regression functional is well defined at some particular model there will be other models arbitrarily close in the metric $d_{\mathcal{H}}$ where a unique solution does not exist. This points to an inherent local instability of high breakdown regression functionals which has been noted in the literature (Sheather et al. (1997); Ellis (1998)). Dietel (1993) has constructed regression functionals which are well defined at all models P with $\Delta(P) < 1$ and which are locally uniformly Lipschitz, not however locally uniformly Fréchet differentiable. For this reason all confidence regions and efficiency claims must be treated with a degree of caution. An increase in stability can however be attained by using the LTS-functional instead of the LMS-functional, by reweighting the observations or using some form of one-step M-functional improvement as in (9.29).

Just as with high breakdown location and scatter functionals in \mathbb{R}^k the calculation of high breakdown regression functionals poses considerable difficulties. The first high breakdown regression functional was Hampel's least median of squares and even in the simplest case of a straight line in \mathbb{R}^2 the computational cost is of order n^2 . The algorithm is by no means simple requiring as it does ideas from computational geometry (see Edelsbrunner and Souvaine (1990)). From this and the fact that the computational complexity increases with dimension it follows that one has to fall back on heuristic algorithms. The one recommended for linear regression is that of Rousseeuw and Van Driessen (1999) for the LTS-functional.

9.4.5 Outliers

To apply the concept of α -outlier regions to the linear regression model we have to specify the distribution P_Y of the response and the joint distribution P_X of the regressors assuming them to be random. For specificity we consider the model

$$P_{Y|X=x} = N(\mathbf{x}^\top \boldsymbol{\beta}, \sigma^2), \quad (9.118)$$

and

$$P_X = \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}). \quad (9.119)$$

Assumption (9.118) states that the conditional distribution of the response given the regressors is normal and assumption (9.119) means that the joint distribution of the regressors is a certain p -variate normal distribution. If both assumptions are fulfilled then the joint distribution of (Y, \mathbf{X}) is a multivariate normal distribution.

We can define outlier regions under model (9.101) in several reasonable ways. If only (9.118) is assumed then a response- α -outlier region could be defined as

$$\text{out}(\alpha, P_{Y|X=x}) = \{y \in \mathbb{R}: u = |y - \mathbf{x}^\top \boldsymbol{\beta}| > \sigma z_{1-\alpha/2}\}, \quad (9.120)$$

which is appropriate if the regressors are fixed and only outliers in y -direction are to be identified. If the regressors are random, which will be the more frequent case in actuarial or econometric applications, outliers in \mathbf{x} -direction are important as well. Under assumption (9.119) a regressor- α -outlier region is a special case of the α -outlier region (9.99). This approach leads to a population based version of the concept of leverage points. These are the points in a sample (y_i, \mathbf{x}_i) , $i = 1, \dots, n$, from model (9.101) “for which \mathbf{x}_i is far away from the bulk of the \mathbf{x}_i in the data” (Rousseeuw and van Zoomeeren (1990)).

For the identification of regressor-outliers (leverage points) the same identification rules can be applied as in the multivariate normal situation. For the detection of response-outliers by resistant one-step identifiers, one needs robust estimators of the regression coefficients and the scale σ . Examples of high breakdown estimators that can be used in this context are the Least Trimmed Squares estimator and the corresponding scale estimator (Rousseeuw (1984); Rousseeuw and Leroy (1987)), S-estimators (Rousseeuw and Yohai (1984)), MM-estimators (Yohai (1987)) or the REWLS-estimators (Gervini and Yohai (2002)).

9.5 Analysis of Variance

9.5.1 One-way Table

The one-way analysis of variance is concerned with the comparison of the locations of k samples x_{ij} , $j = 1, \dots, n_i$, $i = 1, \dots, k$. The term “analysis of variance” goes

back to the pioneering work of Fisher (1935) who decomposed the variance of the combined samples as follows

$$\sum_{ij} (x_{ij} - \bar{x})^2 = \sum_i \sum_j (x_{ij} - \bar{x}_i)^2 + \sum_i n_i (\bar{x}_i - \bar{x})^2. \quad (9.121)$$

The first term of (9.121) is the total sum of squares, the second is the sum of squares within samples and the third is the sum of squares between samples. If the data are modelled as i.i.d. normal random variables with a common variance σ^2 but with the i th sample mean μ_i then it is possible to derive a test for the null hypothesis that the means are equal. The single hypothesis of equal means is rarely of interest in itself. All pairwise comparisons

$$\mu_i = \mu_l, \quad 1 \leq i < l \leq k,$$

as well as contrasts $\sum_i c_i \mu_i = 0$ may also be of interest and give rise to the problem of multiple testing and the associated difficulties. The use of the L_2 -norm as in (9.121) is widespread perhaps because of the elegant mathematics. The peculiarities of data analysis must however have priority over mathematical theory and as real data sets may contain outliers, be skewed to some extent and have different scales it becomes clear that an L_2 -norm and Gaussian based theory is of limited applicability. We sketch a robustified approach to the one-way table (see Davies (2004)).

As a first step gross outliers are eliminated from each sample using a simplified version of the outlier identification rule based on the median and MAD of the sample. Using the robust location and scale functionals T_l and T_s an α_k confidence or approximation interval I_i for location for the i th sample is calculated. To control the error rate for Gaussian and other samples we set $\alpha_k = \alpha^{1/k}$ with for example $\alpha = 0.95$. This choice guarantees that for Gaussian samples

$$P(\mu_i \in I_i, i = 1, \dots, k) = \alpha. \quad (9.122)$$

Simulations show that this holds accurately for other symmetric distributions such as the slash, Cauchy and the double exponential. All questions relating to the locations of the samples are now reduced to questions concerning the intervals. For example, the samples i and l can be approximated by the same location value if and only if $I_i \cap I_l \neq \emptyset$. Similarly if the samples are in some order derived from a covariable it may be of interest as to whether the locations can be taken to be non-decreasing. This will be the case if and only if there exist $a_i, i = 1, \dots, k$ with $a_1 \leq a_2 \leq \dots \leq a_k$ and $a_i \in I_i$ for each i . Because of (9.122) all such questions when stated in terms of the μ_i can be tested simultaneously and on Gaussian test beds the error rate will be $1 - \alpha$ regardless of the number of tests. Another advantage of the method is that it allows a graphical representation. Every analysis should include a plot of the boxplots for the k data sets. This can be augmented by the corresponding plot of the intervals I_i which will often look like the boxplots but if the sample sizes differ greatly this will influence the lengths of the intervals but not the form of the boxplots.

9.5.2 Two-way Table

Given IJ samples

$$(x_{ijk})_{k=1}^{n_{ij}}, \quad i = 1, \dots, I, j = 1, \dots, J$$

the two-way analysis of variance in its simplest version looks for a decomposition of the data of the form

$$x_{ijk} = m + a_i + b_j + c_{ij} + r_{ijk} \quad (9.123)$$

with the the following interpretation. The overall effect is represented by m , the row and column effects by the a_i and b_j respectively and the interactions by the c_{ij} . The residuals r_{ijk} take care of the rest. As it stands the decomposition (9.123) is not unique but can be made so by imposing side conditions on the a_i , b_j and the c_{ij} . Typically these are of the form

$$\sum_i a_i = \sum_j b_j = \sum_i c_{ij} = \sum_j c_{ij} = 0, \quad (9.124)$$

where the latter two hold for all j and i respectively. The conditions (9.124) are almost always stated as technical conditions required to make the decomposition (9.123) identifiable. The impression is given that they are neutral with respect to any form of data analysis. But this is not the case as demonstrated by Tukey (1993) and as can be seen by considering the restrictions on the interactions c_{ij} . The minimum number of interactions for which the restrictions hold is four which, in particular, excludes the case of a single interaction in one cell. The restrictions on the row and column effects can also be criticized but we take this no further than mentioning that the restrictions

$$\text{MED}(a_1, \dots, a_I) = \text{MED}(b_1, \dots, b_J) = 0 \quad (9.125)$$

may be more appropriate. The following robustification of the two-way table is based on Terbeck and Davies (1998). The idea is to look for a decomposition which minimizes the number of non-zero interactions. We consider firstly the case of one observation per cell, $n_{ij} = 1$, for all i and j , and look for a decomposition

$$x_{ij} = m + a_i + b_j + c_{ij} \quad (9.126)$$

with the smallest number of c_{ij} which are non-zero. We denote the positions of the c_{ij} by a $I \times J$ -matrix C with $C(i, j) = 1$ if and only if $c_{ij} \neq 0$, the remaining entries being zero. It can be shown that for certain matrices C the non-zero interactions c_{ij} can be recovered whatever their values and, moreover, they are the unique non-zero residuals of the L_1 -minimization problem

$$\min_{a_i, b_j} \sum_{ij} |x_{ij} - a_i - b_j|. \quad (9.127)$$

We call matrices C for which this holds unconditionally identifiable. They can be characterized and two such matrices are

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \quad (9.128)$$

as well as matrices obtained from any permutations of rows and columns. The above considerations apply to exact models without noise. It can be shown however that the results hold true if noise is added in the sense that for unconditionally identifiable matrices sufficiently large (compared to the noise) interactions c_{ij} can be identified as the large residuals from an L_1 -fit. Three further comments are in order. Firstly Tukey's median polish can often identify interactions in the two-way-table. This is because it attempts to approximate the L_1 -solution. At each step the L_1 -norm is reduced or at least not increased but unfortunately the median polish may not converge and, even if it does, it may not reach the L_1 solution. Secondly L_1 solutions in the presence of noise are not unique. This can be overcome by approximating the modulus function $|x|$ by a strictly convex function almost linear in the tails. Thirdly, if there is more than one observation per cell it is recommended that they are replaced by the median and the method applied to the medians. Finally we point out that an interaction can also be an outlier. There is no a priori way of distinguishing the two.

References

- Adrover, J. (1998). Minimax bias-robust estimation of the dispersion matrix of multivariate distributions. *Annals of Statistics*, 26:2301–2320.
- Andrews, D.F., Bickel, P.J., Hampel, F.R., Rogers, W.H., and Tukey, J.W. (1972). *Robust Estimates of Location: Survey and Advances*. Princeton University Press, Princeton, N.J.
- Atkinson, A.C. (1994). Fast very robust methods for the detection of multiple outliers. *Journal of the American Statistical Association*, 89:1329–1339.
- Barme-Delcroix, M.-F. and Gather, U. (2000). An isobar-surfaces approach to multi-dimensional outlier-proneness. Technical Report 20, Sonderforschungsbereich 475, University of Dortmund, Dortmund, Germany.
- Barnett, V. and Lewis, T. (1994). *Outliers in Statistical Data*. Wiley, New York, third edition.
- Bartlett, M.S. (1935). The effect of non-normality on the t -distribution. *Proceedings of the Cambridge Philosophical Society*, 31:223–231.
- Becker, C. and Gather, U. (1999). The masking breakdown point of multivariate outlier identification rules. *Journal of the American Statistical Association*, 94:947–955.

- Becker, C. and Gather, U. (2001). The largest nonidentifiable outlier: a comparison of multivariate simultaneous outlier identification rules. *Computational Statistics and Data Analysis*, 36:119–127.
- Bednarski, T. (1993). Fréchet differentiability and robust estimation. In Mandl, P. and Husková, M. (eds), *Asymptotic Statistics: Proceedings of the Fifth Prague Symposium*, Springer Lecture Notes, pp. 49–58. Springer.
- Bednarski, T. and Clarke, B.R. (1998). On locally uniform expansions of regular functionals. *Discussiones Mathematicae: Algebra and Stochastic Methods*, 18:155–165.
- Bednarski, T., Clarke, B.R., and Kolkiewicz, W. (1991). Statistical expansions and locally uniform Fréchet differentiability. *Journal of the Australian Mathematical Society*, 50:88–97.
- Bernholt, T. and Fischer, P. (2001). The complexity of computing the mcd-estimator. Technical Report 45, Sonderforschungsbereich 475, University of Dortmund, Dortmund, Germany.
- Berrendero, J.R. and Zamar, R.H. (2001). Maximum bias curves for robust regression with non-elliptical regressors. *Annals of Statistics*, 29:224–251.
- Box, G.E.P. (1953). Non-normality and test on variance. *Biometrika*, 40:318–335.
- Box, G.E.P. and Andersen, S.L. (1955). Permutation theory in the derivation of robust criteria and the study of departures from assumption. *Journal of the Royal Statistical Society Series B*, 17:1–34.
- Caroni, C. and Prescott, P. (1992). Sequential application of wilk's multivariate outlier test. *Applied Statistics*, 41:355–364.
- Chang, H., McKean, J.W., Narjano, J.D., and Sheather, S.J. (1999). High-breakdown rank regression. *Journal of the American Statistical Association*, 94(445):205–219.
- Clarke, B.R. (1983). Uniqueness and Fréchet differentiability of functional solutions to maximum likelihood type equations. *Annals of Statistics*, 11:1196–1205.
- Cohen, M. (1991). The background of configural polysampling: a historical perspective. In Morgenthaler, S. and Tukey, J.W. (eds), *Configural Polysampling: A Route to Practical Robustness*, Chap. 2. Wiley, New York.
- Croux, C. and Dehon, C. (2001). Robust linear discriminant analysis using S-estimators. *Canadian Journal of Statistics*, 29:473–492.
- Croux, C. and Haesbroeck, G. (2000). Principal components analysis based on robust estimators of the covariance or correlation matrix: influence functions and efficiencies. *Biometrika*, 87:603–618.
- Croux, C. and Rousseeuw, P.J. (1992). Time-efficient algorithms for two highly robust estimators of scale. In Dodge, Y. and Whittaker, J.C. (eds), *Computational Statistics*, volume 1, pp. 411–428, Heidelberg. Physica-Verlag.
- Davies, P.L. (1987). Asymptotic behaviour of S-estimates of multivariate location parameters and dispersion matrices. *Annals of Statistics*, 15:1269–1292.
- Davies, P.L. (1992a). The asymptotics of Rousseeuw's minimum volume ellipsoid. *Annals of Statistics*, 20:1828–1843.
- Davies, P.L. (1993). Aspects of robust linear regression. *Annals of Statistics*, 21:1843–1899.

- Davies, P.L. (1995). Data features. *Statistica Neerlandica*, 49:185–245.
- Davies, P.L. (1998). On locally uniformly linearizable high breakdown location and scale functionals. *Annals of Statistics*, 26:1103–1125.
- Davies, P.L. (2004). The one-way table. *Journal of Statistical Planning and Inference*, 122:3–13.
- Davies, P.L. and Gather, U. (1993). The identification of multiple outliers (with discussion). *Journal of the American Statistical Association*, 88:782–801.
- Dietel, G. (1993). *Global location and dispersion functionals*. PhD thesis, University of Essen.
- Donoho, D.L. (1982). *Breakdown properties of multivariate location estimators*. PhD thesis, Department of Statistics, Harvard University, Harvard, Mass.
- Donoho, D.L. and Gasko, M. (1992). Breakdown properties of location estimates based on halfspace depth and project outlyingness. *Annals of Statistics*, 20:1803–1827.
- Donoho, D.L. and Huber, P.J. (1983). The notion of breakdown point. In Bickel, P.J., Doksum, K.A. and Hodges, J.L. Jr. (eds), *A Festschrift for Erich L. Lehmann*, pp. 157–184, Belmont, California. Wadsworth.
- Eddington, A.S. (1914). *Stellar Movements and the Structure of the Universe*. Macmillan, New York.
- Edelsbrunner, H. and Souvaine, D. (1990). Computing median-of-squares regression lines and guided topological sweep. *Journal of the American Statistical Association*, 85:115–119.
- Ellis, S.P. (1998). Instability of least squares, least absolute deviation and least median of squares linear regression. *Statistical Science*, 13(4):337–350.
- Fernholz, L.T. (1983). *Von Mises Calculus for Statistical Functionals*. Number 19 in Lecture Notes in Statistics. Springer-Verlag, New York.
- Fisher, R.A. (1920). A mathematical examination of the methods of determining the accuracy of an observation by the mean error and the mean square error. *Monthly Notices of the Royal Astronomical Society*, 80:758–770.
- Fisher, R.A. (1935). *The Design of Experiments*, Oliver and Boyd, Edinburgh and London.
- Gather, U. (1990). Modelling the occurrence of multiple outliers. *Allgemeines Statistisches Archiv*, 74:413–428.
- Gather, U. and Hilker, T. (1997). A note on tyler's modification of the MAD for the stahel-donoho estimator. *Annals of Statistics*, 25:2024–2026.
- Gather, U., Kuhnt, S., and Pawlitschko, J. (2003). Concepts of outlyingness for various data structures. In Misra, J.C. (ed), *Industrial Mathematics and Statistics*. Narosa Publishing House, New Delhi, 545–585.
- Gather, U. and Schultze, V. (1999). Robust estimation of scale of an exponential distribution. *Statistica Neerlandica*, 53:327–341.
- Gayen, A.K. (1950). The distribution of the variance ratio in random samples of any size drawn from non-normal universe. *Biometrika*, 37:236–255.
- Geary, R.C. (1936). The distribution of 'student's' ratio for non-normal samples. *Journal of the Royal Statistical Society Supplement*, 3:178–184.
- Geary, R.C. (1947). Testing for normality. *Biometrika*, 34:209–242.

- Gervini, D. and Yohai, V.J. (2002). A class of robust and fully efficient regression estimators. *Annals of Statistics*, 30(2):583–616.
- Gnanadesikan, R. and Kettenring, J.R. (1972). Robust estimates, residuals, and outlier detection with multiresponse data. *Biometrics*, 28:81–124.
- Hadi, A.S. (1994). A modification of a method for the detection of outliers in multivariate samples. *Journal of the Royal Statistical Society, Series B*, 56:393–396.
- Hadi, A.S. and Simonoff, J.S. (1997). Procedures for the identification of multiple outliers in linear models. *Journal of the American Statistical Association*, 88:1264–1272.
- Hampel, F.R. (1968). *Contributions to the theory of robust estimation*. PhD thesis, University of California, Berkeley.
- Hampel, F.R. (1975). Beyond location parameters: Robust concepts and methods (with discussion). In *Proceedings of the 40th Session of the ISI*, volume 46, Book 1, pp. 375–391.
- Hampel, F.R. (1985). The breakdown points of the mean combined with some rejection rules. *Technometrics*, 27:95–107.
- Hampel, F.R., Ronchetti, E.M., Rousseeuw, P.J., and Stahel, W.A. (1986). *Robust Statistics: The Approach Based on Influence Functions*, Wiley, New York.
- Hawkins, D.M. (1980). *Identification of outliers*, Chapman and Hall, London.
- Huber, P.J. (1964). Robust estimation of a location parameter. *Annals of Mathematical Statistics*, 35:73–101.
- Huber, P.J. (1977). Robust statistical procedures. In *Regional Conference Series in Applied Mathematics No. 27*, Society for Industrial and Applied Mathematics, Philadelphia, Penn.
- Huber, P.J. (1981). *Robust Statistics*, Wiley, New York.
- Huber, P.J. (1984). Finite sample breakdown points of m - and p -estimators. *Annals of Statistics*, 12:119–126.
- Huber, P.J. (1995). Robustness: Where are we now? *Student*, 1:75–86.
- Kent, J.T. and Tyler, D.E. (1991). Redescending M-estimates of multivariate location and scatter. *Annals of Statistics*, 19:2102–2119.
- Kent, J.T. and Tyler, D.E. (1996). Constrained M-estimation for multivariate location and scatter. *Annals of Statistics*, 24:1346–1370.
- Liu, R.Y., Parelius, J.M., and Singh, K. (1999). Multivariate analysis by data depth: descriptive statistics, graphics and inference. *Annals of Statistics*, 27:783–840.
- Lopuhaä, H.P. (1989). On the relation between S-estimators and M-estimators of multivariate location and covariance. *Annals of Statistics*, 19:229–248.
- Lopuhaä, H.P. (1991). Multivariate r -estimators for location and scatter. *Canadian Journal of Statistics*, 19:307–321.
- Lopuhaä, H.P. and Rousseeuw, P.J. (1991). Breakdown properties of affine equivariant estimators of multivariate location and covariance matrices. *Annals of Statistics*, 19:229–248.
- Marazzi, A. (1992). *Algorithms, Routines, and S-Functions for Robust Statistics*. Chapman and Hall, New York.

- Maronna, R.A. (1976). Robust M -estimators of multivariate location and scatter. *Annals of Statistics*, 4(1):51–67.
- Maronna, R.A., Stahel, W.A., and Yohai, V.J. (1992). Bias-robust estimators of multivariate scatter based on projections. *Journal of Multivariate Analysis*, 42:141–161.
- Maronna, R.A. and Yohai, V.J. (1981). Asymptotic behavior of general M -estimates for regression and scale with random carriers. *Zeitschrift für Wahrscheinlichkeitstheorie und verwandte Gebiete*, 58:7–20.
- Maronna, R.A. and Yohai, V.J. (1993). Bias-robust estimates of regression based on projections. *Annals of Statistics*, 21(2):965–990.
- Martin, R.D., Yohai, V.J., and Zamar, R.H. (1989). Min-max bias robust regression. *Annals of Statistics*, 17(4):1608–1630.
- Martin, R.D. and Zamar, R.H. (1993a). Bias robust estimation of scale. *Annals of Statistics*, 21(2):991–1017.
- Martin, R.D. and Zamar, R.H. (1993b). Efficiency constrained bias robust estimation of location. *Annals of Statistics*, 21(1):338–354.
- Mendes, B. and Tyler, D.E. (1996). Constrained M -estimates for regression. In Rieder, H. (ed), *Robust Statistics; Data Analysis and Computer Intensive Methods*, number 109 in Lecture Notes in Statistics, pp. 299–320. Springer-Verlag.
- Neyman, J. and Pearson, E.S. (1933). On the problem of the most efficient tests of statistical hypotheses. *Philosophical Transactions of the Royal Society (London)*, Series A, 231:289–337.
- Pearson, E.S. (1929). The distribution of frequency constants in small samples from non-normal symmetrical and skew populations. *Biometrika*, 21:259–286.
- Pearson, E.S. (1931). The analysis of variance in cases of non-normal variation. *Biometrika*, 23:114–133.
- Pearson, E.S. and Chandra Sekar, S. (1936). The efficiency of statistical tools and a criterion for the rejection of outlying observations. *Biometrika*, 28:308–320.
- Pollard, D. (1984). *Convergence of stochastic processes*. Springer-Verlag, New York.
- Riedel, M. (1989a). On the bias-robustness in the location model i. *Statistics*, 2:223–233.
- Riedel, M. (1989b). On the bias-robustness in the location model ii. *Statistics*, 2:235–246.
- Rieder, H. (1994). *Robust Asymptotic Statistics*. Springer, Berlin.
- Rocke, D.M. (1996). Robustness properties of S -estimators of multivariate location and shape in high dimension. *Annals of Statistics*, 24:1327–1345.
- Rocke, D.M. and Woodruff, D.L. (1996). Identification of outliers in multivariate data. *Journal of the American Statistical Association*, 91(435):1047–1061.
- Rocke, D.M. and Woodruff, D.L. (1997). Robust estimation of multivariate location and shape. *Journal of Statistical Planning and Inference*, 91:245–255.
- Rosner, B. (1975). On the detection of many outliers. *Technometrics*, 17:221–227.
- Rousseeuw, P.J. (1984). Least median of squares regression. *Journal of the American Statistical Association*, 79:871–880.

- Rousseeuw, P.J. (1985). Multivariate estimation with high breakdown point. In Grossmann, W., Pflug, C.G., Vincze, I. and Wertz, W. (eds), *Mathematical Statistics and Applications (Proceedings of the 4th Pannonian Symposium on Mathematical Statistics)*, volume B, Dordrecht. Reidel.
- Rousseeuw, P.J. and Croux, C. (1992). Explicit scale estimators with high breakdown point. In Dodge, Y. (ed), *L_1 -Statistical Analysis and Related Methods*, pp. 77–92, Amsterdam. North Holland.
- Rousseeuw, P.J. and Croux, C. (1993). Alternatives to the median absolute deviation. *Journal of the American Statistical Association*, 88:1273–1283.
- Rousseeuw, P.J. and Croux, C. (1994). The bias of k-step M-estimators. *Statistics and Probability Letters*, 20:411–420.
- Rousseeuw, P.J. and Hubert, M. (1999). Regression depth. *Journal of the American Statistical Association*, 94:388–402.
- Rousseeuw, P.J. and Leroy, A.M. (1987). *Robust Regression and Outlier Detection*, Wiley, New York.
- Rousseeuw, P.J. and Van Driessen, K. (1999). A fast algorithm for the minimum covariance determinant estimator. *Technometrics*, 41:212–223.
- Rousseeuw, P.J. and Van Driessen, K. (2000). An algorithm for positive-breakdown methods based on concentration steps. In Gaul, W., Opitz, O., and Schader, M. (eds), *Data Analysis: Scientific modelling and Practical Application*, pp. 335–346. Springer-Verlag, New York.
- Rousseeuw, P.J. and van Zoomeeren, B.C. (1990). Unmasking multivariate outliers and leverage points. *Journal of the American Statistical Association*, 85:633–639.
- Rousseeuw, P.J. and Yohai, V.J. (1984). Robust regression by means of S-estimators. In Franke, J. e.a. (ed), *Robust and Nonlinear Time Series Analysis*, pp. 256–272, New York. Springer.
- Scholz, F.W. (1971). *Comparison of optimal location estimators*. PhD thesis, Department of Statistics, University of California, Berkeley.
- Sheather, S.J., McKean, J.W., and Hettmansperger, T.P. (1997). Finite sample stability properties of the least median of squares estimator. *Journal of Statistical Computing and Simulation*, 58(4):371–383.
- Simonoff, J.S. (1984). A comparison of robust methods and detection of outlier techniques when estimating a location parameter. *Communications in Statistics, Series A*, 13:813–842.
- Simonoff, J.S. (1987). The breakdown and influence properties of outlier rejection-plus-mean procedures. *Communications in Statistics, Series A*, 16:1749–1760.
- Stahel, W.A. (1981). Breakdown of covariance estimators. Research Report 31, Fachgruppe für Statistik, ETH, Zurich.
- Staudte, R.G. and Sheather, S.J. (1990). *Robust Estimation and Testing*, Wiley, New York.
- Tatsuoka, K.S. and Tyler, D.E. (2000). On the uniqueness of S-functionals and M-functionals under non-elliptic distributions. *Annals of Statistics*, 28(4):1219–1243.

- Terbeck, W. and Davies, P.L. (1998). Interactions and outliers in the two-way analysis of variance. *Annals of Statistics*, 26:1279–1305.
- Tietjen, G.L. and Moore, R.H. (1972). Some grubbs-type statistics for the detection of several outliers. *Technometrics*, 14:583–597.
- Tukey, J.W. (1960). A survey of sampling from contaminated distributions. In Olkin, I. (ed), *Contributions to Probability and Statistics*. Stanford University Press, Stanford, California.
- Tukey, J.W. (1975). Mathematics and picturing data. In *Proceedings of International Congress of Mathematicians, Vancouver*, volume 2, pp. 523–531.
- Tukey, J.W. (1993). Exploratory analysis of variance as providing examples of strategic choices. In Morgenthaler, S., Ronchetti, E., and Stahel, W.A. (eds), *New Directions in Statistical Data Analysis and Robustness*, Basel. Birkhäuser.
- Tyler, D.E. (1994). Finite sample breakdown points of projection based multivariate location and scatter statistics. *Annals of Statistics*, 22:1024–1044.
- von Mises, R. (1937). Sur les fonctions statistiques. In *Conférence de la Réunion Internationale des Mathématiciens*. Gauthier-Villars.
- Willems, S., Pison, G., Rousseeuw, P.J., and Van Aelst, S. (2002). A robust Hotelling test. *Metrika*, 55:125–138.
- Yohai, V.J. (1987). High breakdown point and high efficiency robust estimates for regression. *Annals of Statistics*, 15:642–656.
- Yohai, V.J. and Maronna, R.A. (1990). The maximum bias of robust covariances. *Communications in Statistics – Theory and Methods*, 19:3925–3933.
- Yohai, V.J. and Zamar, R.H. (1988). High breakdown point estimates of regression by means of the minimization of an efficient scale. *Journal of the American Statistical Association*, 83:406–413.
- Zuo, Y. (2001). Some quantitative relationships between two types of finite sample breakdown points. *Statistics and Probability letters*, 51:369–375.
- Zuo, Y. and Serfling, R. (2000a). General notions of statistical depth function. *Annals of Statistics*, 28:461–482.
- Zuo, Y. and Serfling, R. (2000b). Structural properties and convergence results for contours of sample statistical depth functions. *Annals of Statistics*, 28:483–499.

Semiparametric Models

III.10

Joel L. Horowitz

10.1	<i>Introduction</i>	699
10.2	<i>Semiparametric Models for Conditional Mean Functions</i>	701
	Single Index Models	702
	Partially Linear Models	705
	Nonparametric Additive Models	705
	Transformation Models	707
10.3	<i>The Proportional Hazards Model with Unobserved Heterogeneity</i>	711
10.4	<i>A Binary Response Model</i>	714

10.1

Introduction

Much empirical research is concerned with estimating conditional mean, median, or hazard functions. For example, labor economists are interested in estimating the mean wages of employed individuals conditional on characteristics such as years of work experience and education. The most frequently used estimation methods assume that the function of interest is known up to a set of constant parameters that can be estimated from data. Models in which the only unknown quantities are a finite set of constant parameters are called *parametric*. The use of a parametric model greatly simplifies estimation, statistical inference, and interpretation of the estimation results but is rarely justified by theoretical or other *a priori* considerations. Estimation and inference based on convenient but incorrect assumptions about the form of the conditional mean function can be highly misleading.

As an illustration, the solid line in Fig. 10.1 shows an estimate of the mean of the logarithm of weekly wages, $\log W$, conditional on years of work experience, EXP , for white males with 12 years of education who work full time and live in urban areas of the North Central U.S. The estimate was obtained by applying kernel nonparametric regression (see, e.g., Härdle 1990, Fan and Gijbels 1996) to data from the 1993 Current Population Survey (CPS). The estimated conditional mean of $\log W$ increases steadily up to approximately 30 years of experience and is flat thereafter. The dashed and dotted lines in Fig. 10.1 show two parametric estimates of the mean of the logarithm of weekly wages conditional on years of work experience. The dashed line is the ordinary least squares (OLS) estimate that is obtained by assuming that the mean of $\log W$ conditional on EXP is the linear function $E(\log W|EXP) = \beta_0 + \beta_1 EXP$. The dotted line is the OLS estimate that is obtained by assuming that $E(\log W|EXP)$ is the quadratic function $E(\log W|EXP) = \beta_0 + \beta_1 EXP + \beta_2 EXP^2$. The nonparametric estimate (solid line) places no restrictions

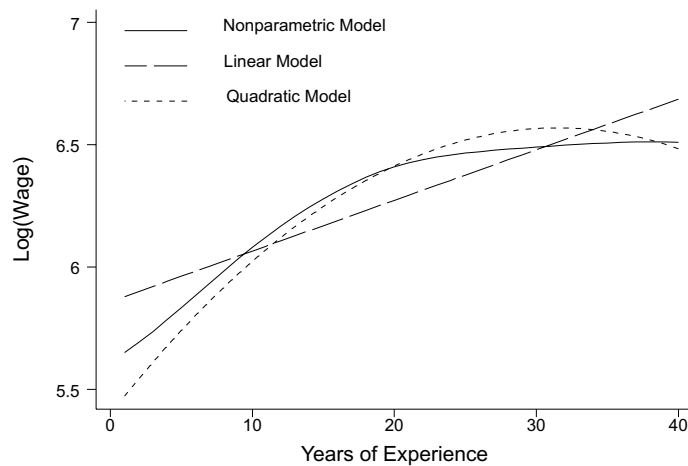


Figure 10.1. Nonparametric and Parametric Estimates of Mean Log Wages

on the shape of $E(\log W|EXP)$. The linear and quadratic models give misleading estimates of $E(\log W|EXP)$. The linear model indicates that $E(\log W|EXP)$ increases steadily as experience increases. The quadratic model indicates that $E(\log W|EXP)$ decreases after 32 years of experience. In contrast, the nonparametric estimate of $E(\log W|EXP)$ becomes nearly flat at approximately 30 years of experience. Because the nonparametric estimate does not restrict the conditional mean function to be linear or quadratic, it is more likely to represent the true conditional mean function.

The opportunities for specification error increase if Y is binary. For example, consider a model of the choice of travel mode for the trip to work. Suppose that the available modes are automobile and transit. Let $Y = 1$ if an individual chooses automobile and $Y = 0$ if the individual chooses transit. Let X be a vector of explanatory variables such as the travel times and costs by automobile and transit. Then $E(Y|x)$ is the probability that $Y = 1$ (the probability that the individual chooses automobile) conditional on $X = x$. This probability will be denoted $P(Y = 1|x)$. In applications of binary response models, it is often assumed that $P(Y = 1|x) = G(\beta'x)$, where β is a vector of constant coefficients and G is a known probability distribution function. Often, G is assumed to be the cumulative standard normal distribution function, which yields a *binary probit* model, or the cumulative logistic distribution function, which yields a *binary logit* model. The coefficients β can then be estimated by the method of maximum likelihood (Amemiya 1985). However, there are now two potential sources of specification error. First, the dependence of Y on x may not be through the linear *index* $\beta'x$. Second, even if the index $\beta'x$ is correct, the *response function* G may not be the normal or logistic distribution function. See Horowitz (1993a, 1998) for examples of specification errors in binary response models and their consequences.

Many investigators attempt to minimize the risk of specification error by carrying out a *specification search* in which several different models are estimated and conclusions are based on the one that appears to fit the data best. Specification searches may be unavoidable in some applications, but they have many undesirable properties and their use should be minimized. There is no guarantee that a specification search will include the correct model or a good approximation to it. If the search includes the correct model, there is no guarantee that it will be selected by the investigator's model selection criteria. Moreover, the search process invalidates the statistical theory on which inference is based.

The rest of this chapter describes methods that deal with the problem of specification error by relaxing the assumptions about functional form that are made by parametric models. The possibility of specification error can be essentially eliminated through the use of nonparametric estimation methods. They assume that the function of interest is smooth but make no other assumptions about its shape or functional form. However, nonparametric methods have important disadvantages that seriously limit their usefulness in applications. One important problem is that the precision of a nonparametric estimator decreases rapidly as the dimension of the explanatory variable X increases. This phenomenon is called the *curse of dimensionality*. As a result of it, impracticably large samples are usually needed to obtain acceptable estimation precision if X is multidimensional, as it often is

in applications. For example, a labor economist may want to estimate mean log wages conditional on years of work experience, years of education, and one or more indicators of skill levels, thus making the dimension of X at least 3.

Another problem is that nonparametric estimates can be difficult to display, communicate, and interpret when X is multidimensional. Nonparametric estimates do not have simple analytic forms. If X is one- or two-dimensional, then the estimate of the function of interest can be displayed graphically as in Fig. 10.1, but only reduced-dimension projections can be displayed when X has three or more components. Many such displays and much skill in interpreting them can be needed to fully convey and comprehend the shape of an estimate.

A further problem with nonparametric estimation is that it does not permit extrapolation. For example, in the case of a conditional mean function it does not provide predictions of $E(Y|x)$ at points x that are outside of the support (or range) of the random variable X . This is a serious drawback in policy analysis and forecasting, where it is often important to predict what might happen under conditions that do not exist in the available data. Finally, in nonparametric estimation, it can be difficult to impose restrictions suggested by economic or other theory. Matzkin (1994) discusses this issue.

Semiparametric methods offer a compromise. They make assumptions about functional form that are stronger than those of a nonparametric model but less restrictive than the assumptions of a parametric model, thereby reducing (though not eliminating) the possibility of specification error. Semiparametric methods permit greater estimation precision than do nonparametric methods when X is multidimensional. They are easier to display and interpret than nonparametric ones and provide limited capabilities for extrapolation and imposing restrictions derived from economic or other theory models. Section 10.2 of this chapter describes some semiparametric models for conditional mean functions. Section 10.3 describes semiparametric estimators for an important class of hazard models. Section 10.4 is concerned with semiparametric estimation of a certain binary response model.

Semiparametric Models for Conditional Mean Functions

10.2

The term *semiparametric* refers to models in which there is an unknown function in addition to an unknown finite dimensional parameter. For example, the binary response model $P(Y = 1|x) = G(\beta'x)$ is semiparametric if the function G and the vector of coefficients β are both treated as unknown quantities. This section describes two semiparametric models of conditional mean functions that are important in applications. The section also describes a related class of models that has no unknown finite-dimensional parameters but, like semiparametric models, mitigates the disadvantages of fully nonparametric models. Finally, this section

describes a class of transformation models that is important in estimation of hazard functions among other applications. Powell (1994) discusses additional semiparametric models.

Single Index Models

10.2.1

In a semiparametric single index model, the conditional mean function has the form

$$E(Y|x) = G(\beta'x) , \quad (10.1)$$

where β is an unknown constant vector and G is an *unknown* function. The quantity $\beta'x$ is called an *index*. The inferential problem is to estimate G and β from observations of (Y, X) . G in (10.1) is analogous to a link function in a generalized linear model, except in (10.1) G is unknown and must be estimated.

Model (10.1) contains many widely used parametric models as special cases. For example, if G is the identity function, then (10.1) is a linear model. If G is the cumulative normal or logistic distribution function, then (10.1) is a binary probit or logit model. When G is unknown, (10.1) provides a specification that is more flexible than a parametric model but retains many of the desirable features of parametric models, as will now be explained.

One important property of single index models is that they avoid the curse of dimensionality. This is because the index $\beta'x$ aggregates the dimensions of x , thereby achieving *dimension reduction*. Consequently, the difference between the estimator of G and the true function can be made to converge to zero at the same rate that would be achieved if $\beta'x$ were observable. Moreover, β can be estimated with the same rate of convergence that is achieved in a parametric model. Thus, in terms of the rates of convergence of estimators, a single index model is as accurate as a parametric model for estimating β and as accurate as a one-dimensional nonparametric model for estimating G . This dimension reduction feature of single index models gives them a considerable advantage over nonparametric methods in applications where X is multidimensional and the single index structure is plausible.

A single-index model permits limited extrapolation. Specifically, it yields predictions of $E(Y|x)$ at values of x that are not in the support of X but are in the support of $\beta'X$. Of course, there is a price that must be paid for the ability to extrapolate. A single index model makes assumptions that are stronger than those of a nonparametric model. These assumptions are testable on the support of X but not outside of it. Thus, extrapolation (unavoidably) relies on untestable assumptions about the behavior of $E(Y|x)$ beyond the support of X .

Before β and G can be estimated, restrictions must be imposed that insure their identification. That is, β and G must be uniquely determined by the population distribution of (Y, X) . Identification of single index models has been investigated by Ichimura (1993) and, for the special case of binary response models, Manski (1988). It is clear that β is not identified if G is a constant function or there is an exact

linear relation among the components of X (perfect multicollinearity). In addition, (10.1) is observationally equivalent to the model $E(Y|X) = G^*(\gamma + \delta\beta'x)$, where γ and $\delta \neq 0$ are arbitrary and G^* is defined by the relation $G^*(\gamma + \delta v) = G(v)$ for all v in the support of $\beta'X$. Therefore, β and G are not identified unless restrictions are imposed that uniquely specify γ and δ . The restriction on γ is called *location normalization* and can be imposed by requiring X to contain no constant (intercept) component. The restriction on δ is called *scale normalization*. Scale normalization can be achieved by setting the β coefficient of one component of X equal to one. A further identification requirement is that X must include at least one continuously distributed component whose β coefficient is non-zero. Horowitz (1998) gives an example that illustrates the need for this requirement. Other more technical identification requirements are discussed by Ichimura (1993) and Manski (1988).

The main estimation challenge in single index models is estimating β . Given an estimator b_n of β , G can be estimated by carrying out the nonparametric regression of Y on $b_n'X$ (e.g. by using kernel estimation). Several estimators of β are available. Ichimura (1993) describes a nonlinear least squares estimator. Klein and Spady (1993) describe a semiparametric maximum likelihood estimator for the case in which Y is binary. These estimators are difficult to compute because they require solving complicated nonlinear optimization problems. Powell, *et al.* (1989) describe a *density-weighted average derivative estimator* (DWADE) that is non-iterative and easily computed. The DWADE applies when all components of X are continuous random variables. It is based on the relation

$$\beta \propto E [p(X)\partial G(\beta'X)/\partial X] = -2E [Y\partial p(X)/\partial X] , \quad (10.2)$$

where p is the probability density function of X and the second equality follows from integrating the first by parts. Thus, β can be estimated up to scale by estimating the expression on the right-hand side of the second equality. Powell, *et al.* (1989) show that this can be done by replacing p with a nonparametric estimator and replacing the population expectation E with a sample average. Horowitz and Härdle (1996) extend this method to models in which some components of X are discrete. Hristache, Juditsky, and Spokoiny (2001) developed an iterated average derivative estimator that performs well when X is high-dimensional. Ichimura and Lee (1991) and Hristache, Juditsky, Polzehl and Spokoiny (2001) investigate multiple-index generalizations of (10.1).

The usefulness of single-index models can be illustrated with an example that is taken from Horowitz and Härdle (1996). The example consists of estimating a model of product innovation by German manufacturers of investment goods. The data, assembled in 1989 by the IFO Institute of Munich, consist of observations on 1100 manufacturers. The dependent variable is $Y = 1$ if a manufacturer realized an innovation during 1989 in a specific product category and 0 otherwise. The independent variables are the number of employees in the product category (*EMPLP*), the number of employees in the entire firm (*EMPLF*), an indicator of the firm's production capacity utilization (*CAP*), and a variable *DEM*, which is 1 if a firm expected increasing demand in the product category and 0 otherwise.

The first three independent variables are standardized so that they have units of standard deviations from their means. Scale normalization was achieved by setting $\beta_{EMPLP} = 1$.

Table 10.1 shows the parameter estimates obtained using a binary probit model and the semiparametric method of Horowitz and Härdle (1996). Figure 10.2 shows a kernel estimate of $G'(v)$. There are two important differences between the semiparametric and probit estimates. First, the semiparametric estimate of β_{EMPLF} is

Table 10.1. Estimated Coefficients (Standard Errors) for Model of Product Innovation

EMPLP	EMPLF	CAP	DEM
Semiparametric Model			
1	0.032 (0.023)	0.346 (0.078)	1.732 (0.509)
Probit Model			
1	0.516 (0.024)	0.520 (0.163)	1.895 (0.387)

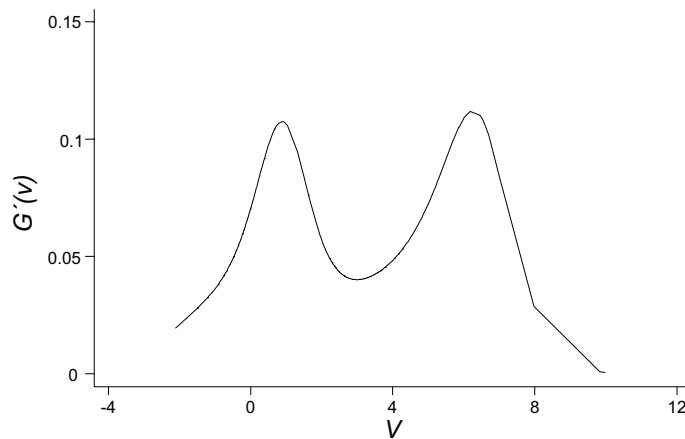


Figure 10.2. Estimate of $G'(v)$ for model of product innovation

small and statistically nonsignificant, whereas the probit estimate is significant at the 0.05 level and similar in size to β_{CAP} . Second, in the binary probit model, G is a cumulative normal distribution function, so G' is a normal density function. Figure 10.2 reveals, however, that G' is bimodal. This bimodality suggests that the data may be a mixture of two populations. An obvious next step in the analysis of the data would be to search for variables that characterize these populations. Standard diagnostic techniques for binary probit models would provide no indication that G' is bimodal. Thus, the semiparametric estimate has revealed an important feature of the data that could not easily be found using standard parametric methods.

10.2.2 Partially Linear Models

In a partially linear model, X is partitioned into two non-overlapping subvectors, X_1 and X_2 . The model has the form

$$E(Y|x_1, x_2) = \beta'x_1 + G(x_2), \quad (10.3)$$

where β is an unknown constant vector and G is an unknown function. This model is distinct from the class of single index models. A single index model is not partially linear unless G is a linear function. Conversely, a partially linear model is a single index model only in this case. Stock (1989, 1991) and Engle *et al.* (1986) illustrate the use of (10.3) in applications. Identification of β requires the *exclusion restriction* that none of the components of X_1 are perfectly predictable by components of X_2 . When β is identified, it can be estimated with an $n^{-1/2}$ rate of convergence regardless of the dimensions of X_1 and X_2 . Thus, the curse of dimensionality is avoided in estimating β .

An estimator of β can be obtained by observing that (10.3) implies

$$Y - E(Y|x_2) = \beta' [X_1 - E(X_1|x_2)] + U, \quad (10.4)$$

where U is an unobserved random variable satisfying $E(U|x_1, x_2) = 0$. Robinson (1988) shows that under regularity conditions, β can be estimated by applying OLS to (10.4) after replacing $E(Y|x_2)$ and $E(X_1|x_2)$ with nonparametric estimators. The estimator of β , b_n , converges at rate $n^{-1/2}$ and is asymptotically normally distributed. G can be estimated by carrying out the nonparametric regression of $Y - b_n'X_1$ on X_2 . Unlike b_n , the estimator of G suffers from the curse of dimensionality; its rate of convergence decreases as the dimension of X_2 increases.

10.2.3 Nonparametric Additive Models

Let X have d continuously distributed components that are denoted X_1, \dots, X_d . In a nonparametric additive model of the conditional mean function,

$$E(Y|x) = \mu + f_1(x_1) + \dots + f_d(x_d), \quad (10.5)$$

where μ is a constant and f_1, \dots, f_d are unknown functions that satisfy a location normalization condition such as

$$\int f_k(v)w_k(v)dv = 0, \quad k = 1, \dots, d, \quad (10.6)$$

where w_k is a non-negative weight function. An additive model is distinct from a single index model unless $E(Y|x)$ is a linear function of x . Additive and partially linear models are distinct unless $E(Y|x)$ is partially linear and G in (10.3) is additive.

An estimator of f_k ($k = 1, \dots, d$) can be obtained by observing that (10.5) and (10.6) imply

$$f_k(x_k) = \int E(Y|x)w_{-k}(x_{-k})dx_{-k}, \quad (10.7)$$

where x_{-k} is the vector consisting of all components of x except the k 'th and w_{-k} is a weight function that satisfies $\int w_{-k}(x_{-k})dx_{-k} = 1$. The estimator of f_k is obtained by replacing $E(Y|x)$ on the right-hand side of (10.7) with nonparametric estimators. Linton and Nielsen (1995) and Linton (1997) present the details of the procedure and extensions of it. Under suitable conditions, the estimator of f_k converges to the true f_k at rate $n^{-2/5}$ regardless of the dimension of X . Thus, the additive model provides dimension reduction. It also permits extrapolation of $E(Y|x)$ within the rectangle formed by the supports of the individual components of X . Mammen, Linton, and Nielsen (1999) describe a backfitting procedure that is likely to be more precise than the estimator based on (10.7) when d is large. See Hastie and Tibshirani (1990) for an early discussion of backfitting.

Linton and Härdle (1996) describe a generalized additive model whose form is

$$E(Y|x) = G[\mu + f_1(x_1) + \dots + f_d(x_d)], \quad (10.8)$$

where f_1, \dots, f_d are unknown functions and G is a known, strictly increasing (or decreasing) function. Horowitz (2001) describes a version of (10.8) in which G is unknown. Both forms of (10.8) achieve dimension reduction. When G is unknown, (10.8) nests additive and single index models and, under certain conditions, partially linear models.

The use of the nonparametric additive specification (10.5) can be illustrated by estimating the model $E(\log W|EXP, EDUC) = \mu + f_{EXP}(EXP) + f_{EDUC}(EDUC)$, where W and EXP are defined as in Sect. 10.1, and $EDUC$ denotes years of education. The data are taken from the 1993 CPS and are for white males with 14 or fewer years of education who work full time and live in urban areas of the North Central U.S. The results are shown in Fig. 10.3. The unknown functions f_{EXP} and f_{EDUC} are estimated by the method of Linton and Nielsen (1995) and are normalized so that $f_{EXP}(2) = f_{EDUC}(5) = 0$. The estimates of f_{EXP} (Fig. 10.3a) and f_{EDUC} (Fig. 10.3b) are nonlinear and differently shaped. Functions f_{EXP} and f_{EDUC} with different shapes cannot be produced by a single index model, and a lengthy specification search might be needed to find a parametric model that produces the shapes shown in Fig. 10.3. Some of the fluctuations of the estimates of f_{EXP} and f_{EDUC} may be artifacts of random sampling error rather than features of $E(\log W|EXP, EDUC)$. However, a more elaborate analysis that takes account of the effects of random sampling error rejects the hypothesis that either function is linear.

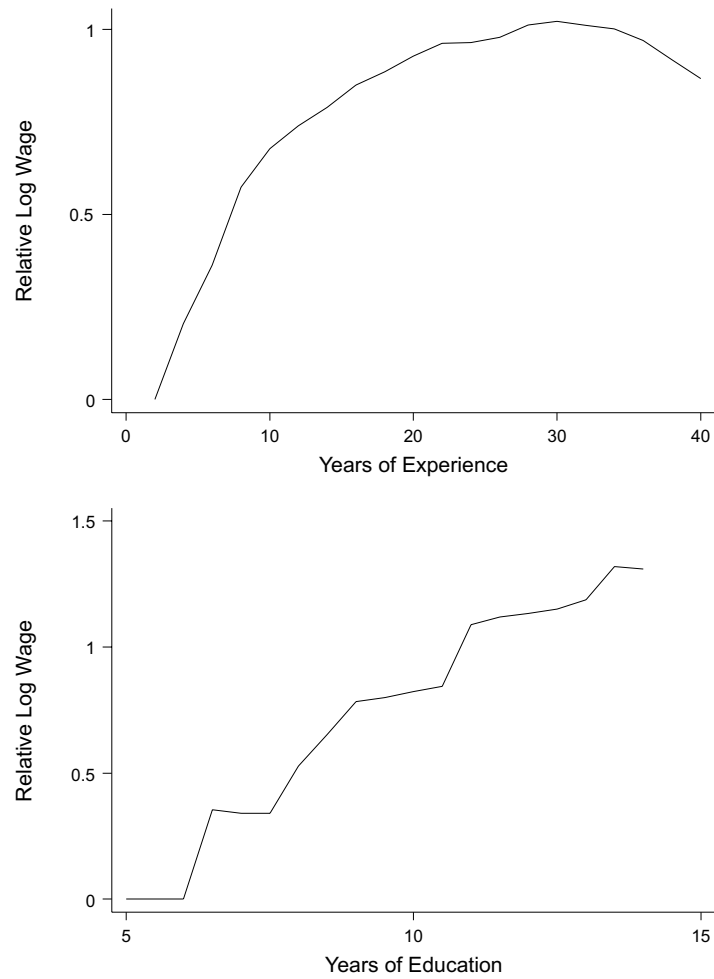


Figure 10.3. Components of nonparametric, additive wage equation

10.2.4 Transformation Models

A transformation model has the form

$$H(Y) = \beta'X + U, \quad (10.9)$$

where H is an unknown increasing function, β is an unknown finite dimensional vector of constants, and U is an unobserved random variable. It is assumed here that U is statistically independent of X . The aim is to estimate H and β . One possibility is to assume that H is known up to a finite-dimensional parameter. For

example, H could be the Box-Cox transformation

$$H(y) = \begin{cases} (y^\tau - 1)/\tau & \text{if } \tau > 0 \\ \log y & \text{if } \tau = 0 \end{cases}$$

where τ is an unknown parameter. Methods for estimating transformation models in which H is parametric have been developed by Amemiya and Powell (1981) and Foster, *et al.* (2001) among others.

Another possibility is to assume that H is unknown but that the distribution of U is known. Cheng, Wei, and Ying (1995, 1997) have developed estimators for this version of (10.9). Consider, first, the problem of estimating β . Let F denote the (known) cumulative distribution function (CDF) of U . Let (Y_i, X_i) and (Y_j, X_j) ($i \neq j$) be two distinct, independent observations of (Y, X) . Then it follows from (10.9) that

$$E[I(Y_i > Y_j)|X_i = x_i, X_j = x_j] = P[U_i - U_j > -(x_i - x_j)] . \quad (10.10)$$

Let $G(z) = P(U_i - U_j > z)$ for any real z . Then

$$G(z) = \int_{-\infty}^{\infty} [1 - F(u + z)] dF(u) .$$

G is a known function because F is assumed known. Substituting G into (10.10) gives

$$E[I(Y_i > Y_j)|X_i = x_i, X_j = x_j] = G[-\beta'(x_i - x_j)] .$$

Define $X_{ij} = X_i - X_j$. Then it follows that β satisfies the moment condition

$$E\{w(\beta'X_{ij})X_{ij}[I(Y_i > Y_j) - G(-\beta'X_{ij})]\} = 0 \quad (10.11)$$

where w is a weight function. Cheng, Wei, and Ying (1995) propose estimating β by replacing the population moment condition (10.11) with the sample analog

$$\sum_{i=1}^n \sum_{j=1}^n \{w(b'X_{ij})X_{ij}[I(Y_i > Y_j) - G(-b'X_{ij})]\} = 0 . \quad (10.12)$$

The estimator of β , b_n , is the solution to (10.12). Equation (10.12) has a unique solution if $w(z) = 1$ for all z and the matrix $\sum_i \sum_j X'_{ij}X_{ij}$ is positive definite. It also has a unique solution asymptotically if w is positive everywhere (Cheng, Wei, and Ying 1995). Moreover, b_n converges almost surely to β . Cheng, Wei, and Ying (1995) also give conditions under which $n^{1/2}(b_n - \beta)$ is asymptotically normally distributed with a mean of 0.

The problem of estimating the transformation function is addressed by Cheng, Wei, and Ying (1997). Equation (10.11) implies that for any real y and vector x that is conformable with X , $EI[I(Y \leq y)|X = x] - F[H(y) - \beta'x] = 0$. Cheng, Wei, and

Ying (1997) propose estimating $H(y)$ by the solution to the sample analog of this equation. That is, the estimator $H_n(y)$ solves

$$n^{-1} \sum_{i=1}^n \{I(Y_i \leq y) - F[H_n(y) - b'_n X_i]\} = 0,$$

where b_n is the solution to (10.12). Cheng, Wei, and Ying (1997) show that if F is strictly increasing on its support, then $H_n(y)$ converges to $H(y)$ almost surely uniformly over any interval $[0, t]$ such that $P(Y > t) > 0$. Moreover, $n^{1/2}(H_n - H)$ converges to a mean-zero Gaussian process over this interval.

A third possibility is to assume that H and F are both nonparametric in (10.9). In this case, certain normalizations are needed to make identification of (10.9) possible. First, observe that (10.9) continues to hold if H is replaced by cH , β is replaced by $c\beta$, and U is replaced by cU for any positive constant c . Therefore, a scale normalization is needed to make identification possible. This will be done here by setting $|\beta_1| = 1$, where β_1 is the first component of β . Observe, also, that when H and F are nonparametric, (10.9) is a semiparametric single-index model. Therefore, identification of β requires X to have at least one component whose distribution conditional on the others is continuous and whose β coefficient is non-zero. Assume without loss of generality that the components of X are ordered so that the first satisfies this condition.

It can also be seen that (10.9) is unchanged if H is replaced by $H + d$ and U is replaced by $U + d$ for any positive or negative constant d . Therefore, a location normalization is also needed to achieve identification when F are nonparametric. Location normalization will be carried out here by assuming that $H(y_0) = 0$ for some finite y_0 . With this location normalization, there is no centering assumption on U and no intercept term in X .

Now consider the problem of estimating H , β , and F . Because (10.9) is a single-index model in this case, β can be estimated using the methods described in Sect. 10.2.1. Let b_n denote the estimator of β . One approach to estimating H and F is given by Horowitz (1996). To describe this approach, define $Z = \beta'X$. Let $G(\cdot|z)$ denote the CDF of Y conditional on $Z = z$. Set $G_y(y|z) = \partial G(y|z)/\partial z$ and $G_z(y|z) = \partial G(y|z)/\partial z$. Then it follows from (10.9) that $H'(y) = -G_y(y|z)/G_z(y|z)$ and that

$$H(y) = - \int_{y_0}^y [G_y(v|z)/G_z(v|z)] dv \quad (10.13)$$

for any z such that the denominator of the integrand is non-zero. Now let $w(\cdot)$ be a scalar-valued, non-negative weight function with compact support S_w such that the denominator of $G_z(v|z)$ is bounded away from 0 for all $v \in [y_0, y]$ and $z \in S_w$. Also assume that

$$\int_{S_w} w(z) dz = 1.$$

Then

$$H(y) = - \int_{y_0}^y \int_{S_w} w(z) [G_y(v|z)/G_z(v|z)] dz dv . \quad (10.14)$$

Horowitz (1996) obtains an estimator of H from (10.14) by replacing G_y and G_z by kernel estimators. Specifically, G_y is replaced by a kernel estimator of the probability density function of Y conditional on $b'_n X = z$, and G_z is replaced by a kernel estimator of the derivative with respect to z of the CDF of Y conditional on $b'_n X = z$. Denote these estimators by G_{ny} and G_{nz} . Then the estimator of H is

$$H_n(y) = - \int_{y_0}^y \int_{S_w} w(z) [G_{ny}(v|z)/G_{nz}(v|z)] dz dv . \quad (10.15)$$

Horowitz (1996) gives conditions under which H_n is uniformly consistent for H and $n^{1/2}(H_n - H)$ converges weakly to a mean-zero Gaussian process. Horowitz (1996) also shows how to estimate F , the CDF of U , and gives conditions under which $n^{1/2}(F_n - F)$ converges to a mean-zero Gaussian process, where F_n is the estimator. Gørgens and Horowitz (1999) extend these results to a censored version of (10.9). Integration over z in (10.14) and (10.15) accelerates the convergence of H_n to H . Kernel estimators converge in probability at rates slower than $n^{-1/2}$. Therefore, $G_{ny}(v|z)/G_{nz}(v|z)$ is not $n^{-1/2}$ -consistent for $G_y(v|z)/G_z(v|z)$. However, integration over z and v in (10.15) creates an averaging effect that causes the integral and, therefore, H_n to converge at the rate $n^{-1/2}$. This is the reason for basing the estimator on (10.14) instead of (10.13).

Other estimators of H when and F are both nonparametric have been proposed by Ye and Duan (1997) and Chen (2002). Chen uses a rank-based approach that is in some ways simpler than that of Horowitz (1996) and may have better finite-sample performance. To describe this approach, define $d_{iy} = I(Y_i > y)$ and $d_{iy_0} = I(Y_j > y_0)$. Let $i \neq j$. Then $E(d_{iy} - d_{iy_0} | X_i, X_j) \geq 0$ whenever $Z_i - Z_j \geq H(y)$. This suggests that if β were known, then $H(y)$ could be estimated by

$$H_n(y) = \arg \max_{\tau} \frac{1}{n(n-1)} \sum_{i=1}^n \sum_{\substack{j=1 \\ j \neq i}}^n (d_{iy} - d_{iy_0}) I(Z_i - Z_j \geq \tau) .$$

Since β is unknown, Chen (2002) proposes

$$H_n(y) = \arg \max_{\tau} \frac{1}{n(n-1)} \sum_{i=1}^n \sum_{\substack{j=1 \\ j \neq i}}^n (d_{iy} - d_{iy_0}) I(b'_n X_i - b'_n X_j \geq \tau) .$$

Chen (2002) gives conditions under which H_n is uniformly consistent for H and $n^{1/2}(H_n - H)$ converges to a mean-zero Gaussian process. Chen (2002) also shows how this method can be extended to a censored version of (10.9).

The Proportional Hazards Model with Unobserved Heterogeneity

Let T denote a duration such as that of a spell of employment or unemployment. Let $F(t|x) = P(T \leq t|X = x)$ where X is a vector of covariates. Let $f(t|x)$ denote the corresponding conditional probability density function. The conditional hazard function is defined as

$$\lambda(t|x) = \frac{f(t|x)}{1 - F(t|x)}.$$

This section is concerned with an approach to modeling $\lambda(t|x)$ that is based on the proportional hazards model of Cox (1972).

The proportional hazards model is widely used for the analysis of duration data. Its form is

$$\lambda(t|x) = \lambda_0(t) e^{-x'\beta}, \quad (10.16)$$

where β is a vector of constant parameters that is conformable with X and λ_0 is a non-negative function that is called the baseline hazard function. The essential characteristic of (10.16) that distinguishes it from other models is that $\lambda(t|x)$ is the product of a function of t alone and a function of x alone. Cox (1972) developed a partial likelihood estimator of β and a nonparametric estimator of λ_0 . Tsiatis (1981) derived the asymptotic properties of these estimators.

In the proportional hazards model with unobserved heterogeneity, the hazard function is conditioned on the covariates X and an unobserved random variable U that is assumed to be independent of X . The form of the model is

$$\lambda(t|x, u) = \lambda_0(t) e^{-(\beta'x+u)}, \quad (10.17)$$

where $\lambda(\cdot|x, u)$ is the hazard conditional on $X = x$ and $U = u$. In a model of the duration of employment U might represent unobserved attributes of an individual (possibly ability) that affect employment duration. A variety of estimators of λ_0 and β have been proposed under the assumption that λ_0 or the distribution of U or both are known up to a finite-dimensional parameter. See, for example, Lancaster (1979), Heckman and Singer (1984a), Meyer (1990), Nielsen, et al. (1992), and Murphy (1994, 1995). However, λ_0 and the distribution of U are nonparametrically identified (Elbers and Ridder 1982, Heckman and Singer 1984b), which suggests that they can be estimated nonparametrically.

Horowitz (1999) describes a nonparametric estimator of λ_0 and the density of U in model (10.17). His estimator is based on expressing (10.17) as a type of transformation model. To do this, define the integrated baseline hazard function, Λ_0 by

$$\Lambda_0(t) = \int_0^t \lambda_0(\tau) d\tau.$$

Then it is not difficult to show that (10.17) is equivalent to the transformation model

$$\log \Lambda_0(T) = X'\beta + U + \varepsilon, \tag{10.18}$$

where ε is a random variable that is independent of X and U and has the CDF $F_\varepsilon(y) = 1 - \exp(-e^y)$. Now define $\sigma = |\beta_1|$, where β_1 is the first component of β and is assumed to be non-zero. Then β/σ and $H = \sigma^{-1} \log \Lambda_0$ can be estimated by using the methods of Sect. 10.2.4. Denote the resulting estimators of β/σ and H by α_n and H_n . If σ were known, then β and Λ_0 could be estimated by $b_n = \sigma\alpha_n$ and $\Lambda_{n0} = \exp(\sigma H_n)$. The baseline hazard function λ_0 could be estimated by differentiating Λ_{n0} . Thus, it is necessary only to find an estimator of the scale parameter σ .

To do this, define $Z = \beta'X$, and let $G(\cdot|z)$ denote the CDF of T conditional on $Z = z$. It can be shown that

$$G(t|z) = 1 - \int \exp \left[-\Lambda_0(t) e^{-(\beta'x+u)} \right] dF(u),$$

where F is the CDF of U . Let p denote the probability density function of Z . Define $G_z(t|z) = \partial G(t|z)/\partial z$ and

$$\sigma(t) = \frac{\int G_z(t|z)p(z)^2 dz}{\int G(t|z)p(z)^2 dz}.$$

Then it can be shown using l'Hospital's rule that if $\Lambda_0(t) > 0$ for all $t > 0$, then

$$\sigma = \lim_{t \rightarrow 0} \sigma(t).$$

To estimate σ , let p_n , G_{nz} and G_n be kernel estimators of p , G_z and G , respectively, that are based on a simple random sample of (T, X) . Define

$$\sigma_n(t) = \frac{\int G_{nz}(t|z)p_n(z)^2 dz}{\int G_n(t|z)p_n(z)^2 dz}.$$

Let c, d , and δ be constants satisfying $0 < c < \infty$, $1/5 < d < 1/4$, and $1/(2d) < \delta < 1$. Let $\{t_{n1}\}$ and $\{t_{n2}\}$ be sequences of positive numbers such that $\Lambda_0(t_{n1}) = cn^{-d}$ and $\Lambda_0(t_{n2}) = cn^{-\delta d}$. Then σ is estimated consistently by

$$\sigma_n = \frac{\sigma_n(t_{n1}) - n^{-d(1-\delta)}\sigma_n(t_{n2})}{n^{-d(1-\delta)}}.$$

Horowitz (1999) gives conditions under which $n^{(1-d)/2}(\sigma_n - \sigma)$ is asymptotically normally distributed with a mean of zero. By choosing d to be close to $1/5$, the rate of convergence in probability of σ_n to σ can be made arbitrarily close to $n^{-2/5}$, which is the fastest possible rate (Ishwaran 1996). It follows from an application of the delta method that the estimators of β , Λ_0 , and λ_0 that are given by $b_n = \sigma_n\alpha_n$, $\Lambda_{n0} = \exp(\sigma_n H_n)$, and $\lambda_{n0} = d\Lambda_{n0}/dt$ are also asymptotically normally distributed with means of zero and $n^{-(1-d)/2}$ rates of convergence. The probability density function of U can be estimated consistently by solving the deconvolution problem $W_n = U + \varepsilon$, where $W_n = \log \Lambda_{n0}(T) - X'\beta_n$. Because the distribution of ε is "supersmooth," the resulting rate of convergence of the estimator of the density of U is $(\log n)^{-m}$, where m is the number of times that the density is differentiable.

This is the fastest possible rate. Horowitz (1999) also shows how to obtain data-based values for t_{n1} and t_{n2} and extends the estimation method to models with censoring.

If panel data on (T, X) are available, then Λ_0 can be estimated with a $n^{-1/2}$ rate of convergence, and the assumption of independence of U from X can be dropped. Suppose that each individual in a random sample of individuals is observed for exactly two spells. Let $(T_j, X_j : j = 1, 2)$ denote the values of (T, X) in the two spells. Define $Z_j = \beta'X_j$. Then the joint survivor function of T_1 and T_2 conditional on $Z_1 = z_1$ and $Z_2 = z_2$ is

$$\begin{aligned} S(t_1, t_2 | Z_1, Z_2) &\equiv P(T_1 > t_1, T_2 > t_2 | Z_1, Z_2) \\ &= \int \exp[-\Lambda_0(t_1) e^{z_1+u} - \Lambda_0(t_2) e^{z_2+u}] dP(u | Z_1 = z_1, Z_2 = z_2). \end{aligned}$$

Honoré(1993) showed that

$$R(t_1, t_2 | z_1, z_2) \equiv \frac{\partial S(t_1, t_2 | z_1, z_2) / \partial t_1}{\partial S(t_1, t_2 | z_1, z_2) / \partial t_2} = \frac{\lambda_0(t_1)}{\lambda_0(t_2)} \exp(z_1 - z_2).$$

Adopt the scale normalization

$$\int_{S_T} \frac{w_t(\tau)}{\lambda_0(\tau)} d\tau = 1,$$

where w_t is a non-negative weight function and S_T is its support. Then

$$\lambda_0(t) = \int_{S_T} w_t(\tau) \exp(z_2 - z_1) R(t, \tau | z_2, z_1) d\tau.$$

Now for a weight function w_z with support S_Z , define

$$w(\tau, z_1, z_2) = w_t(\tau) w_z(z_1) w_z(z_2).$$

Then,

$$\lambda_0(t) = \int_{S_T} d\tau \int_{S_Z} dz_1 \int_{S_Z} dz_2 w(\tau, z_1, z_2) \exp(z_2 - z_1) R(t, \tau | z_1, z_2). \quad (10.19)$$

The baseline hazard function can now be estimated by replacing R with an estimator, R_n , in (10.19). This can be done by replacing Z with $X'b_n$, where b_n is a consistent estimator of β such as a marginal likelihood estimator (Chamberlain 1985, Kalbfleisch and Prentice 1980, Lancaster 2000, Ridder and Tunali 1999),

and replacing S with a kernel estimator of the joint survivor function conditional $X'_1 b_n = z_1$ and $X'_2 b_n = z_2$. The resulting estimator of λ_0 is

$$\lambda_{n0}(t) = \int_{S_T} d\tau \int_{S_Z} dz_1 \int_{S_Z} dz_2 w(\tau, z_1, z_2) \exp(z_2 - z_1) R_n(t, \tau | z_1, z_2) .$$

The integrated baseline hazard function is estimated by

$$\Lambda_{n0}(t) = \int_0^t \lambda_{n0}(\tau) d\tau .$$

Horowitz and Lee (2004) give conditions under which $n^{1/2}(\Lambda_{n0} - \Lambda_0)$ converges weakly to a tight, mean-zero Gaussian process. The estimated baseline hazard function λ_{n0} converges at the rate $n^{-q/(2q+1)}$, where $q \geq 2$ is the number of times that λ_0 is continuously differentiable. Horowitz and Lee (2004) also show how to estimate a censored version of the model.

A Binary Response Model

The general binary response model has the form

$$Y = I(\beta'X + U > 0) , \quad (10.20)$$

where U is an unobserved random variable. If the distribution of U is unknown but depends on X only through the index $\beta'X$, then (10.20) is a single-index model, and β can be estimated by the methods described in Sect. 10.2.1. An alternative model that is non-nested with single-index models can be obtained by assuming that $\text{median}(U|X = x) = 0$ for all x . This assumption places only weak restrictions on the relation between X and the distribution of U . Among other things, it accommodates fairly general types of heteroskedasticity of unknown form, including random coefficients. Under median centering, the inferential problem is to estimate β . The response function, $P(Y = 1|X = x)$ is not identified without making assumptions about the distribution of U that are stronger than those needed to identify and estimate β . Without such assumptions, the only restriction on $P(Y = 1|X = x)$ under median centering is that

$$P(Y = 1|X = x) \begin{cases} > 0.5 & \text{if } \beta'x > 0 \\ = 0.5 & \text{if } \beta'x = 0 \\ < 0.5 & \text{if } \beta'x < 0 \end{cases}$$

Manski (1975, 1985) proposed the first estimator of β under median centering. Let the data be the simple random sample $\{Y_i, X_i : i = 1, \dots, n\}$. The estimator is called the *maximum score* estimator and is

$$b_n = \arg \max_{\|b\|=1} n^{-1} \sum_{i=1}^n (2Y_i - 1)I(b'X_i \geq 0), \quad (10.21)$$

where $\|b\|$ denotes the Euclidean norm of the vector b . The restriction $\|b\| = 1$ is a scale normalization. Scale normalization is needed for identification because (10.20) identifies β only up to scale. Manski (1975, 1985) gave conditions under which b_n consistently estimates β . The rate of convergence of b_n and its asymptotic distribution were derived by Cavanagh (1987) and Kim and Pollard (1990). They showed that the rate of convergence in probability of b_n to β is $n^{-1/3}$ and that $n^{1/3}(b_n - \beta)$ converges in distribution to the maximum of a complicated multidimensional stochastic process. The complexity of the limiting distribution of the maximum score estimator limits its usefulness for statistical inference. Delgado, Rodríguez-Póo and Wolf (2001) proposed using subsampling methods to form confidence intervals for β .

The maximum score estimator has a slow rate of convergence and a complicated asymptotic distribution because it is obtained by maximizing a step function. Horowitz (1992) proposed replacing the indicator function in (10.21) by a smooth function. The resulting estimator of β is called the *smoothed maximum score* estimator. Specifically, let K be a smooth function, possibly but not necessarily a distribution function, that satisfies $K(-\infty) = 0$ and $K(\infty) = 1$. Let $\{h_n : n = 1, 2, \dots\}$ be a sequence of strictly positive constants (bandwidths) that satisfies $h_n \rightarrow 0$ as $n \rightarrow \infty$. The smoothed maximum score estimator, b_{ns} , is

$$b_{ns} = \arg \max_{b \in B} \sum_{i=1}^n (2Y_i - 1)K(X_i'b/h_n),$$

where B is a compact parameter set that satisfies the scale normalization $|b_1| = 1$. Horowitz (1992) shows that under assumptions that are stronger than those of Manski (1975, 1985) but still quite weak, $n^r(b_{ns} - \beta)$ is asymptotically normal, where $2/5 \leq r < 1/2$ and the exact value of r depends on the smoothness of the distribution of $X'\beta$ and of $P(Y = 1|X = x)$. Moreover, the smoothed maximum score estimator has the fastest possible rate of convergence under its assumptions (Horowitz 1993b). Monte Carlo evidence suggests that the asymptotic normal approximation can be inaccurate with samples of practical size. However, Horowitz (2002) shows that the bootstrap, which is implemented by sampling the data randomly with replacement, provides asymptotic refinements for tests of hypotheses about β and produces low ERPs for these tests. Thus, the bootstrap provides a practical way to carry out inference with the smoothed maximum score estimator.

Horowitz (1993c) used the smoothed maximum score method to estimate the parameters of a model of the choice between automobile and transit for work trips in the Washington, D.C., area. The explanatory variables are defined in Table 10.2. Scale normalization is achieved by setting the coefficient of DCOST equal to 1. The data consist of 842 observations sampled randomly from the Washington,

D.C., area transportation study. Each record contains information about a single trip to work, including the chosen mode (automobile or transit) and the values of the explanatory variables. Column 2 of Table 10.2 shows the smoothed maximum

Table 10.2. Smoothed Maximum Score Estimates of a Work-Trip Mode-Choice Model

Variable ^a	Estimated Coefficient	Half-Width of Nominal 90% Conf. Interval Based on	
		Asymp. Normal Approximation	Bootstrap
INTRCPT	-1.5761	0.2812	0.7664
AUTOS	2.2418	0.2989	0.7488
DOVTT	0.0269	0.0124	0.0310
DIVTT	0.0143	0.0033	0.0087
DCOST	1.0 ^b		

^a Definitions of variables: INTRCPT: Intercept term equal to 1; AUTOS: Number of cars owned by traveler's household; DOVTT: Transit out-of-vehicle travel time minus automobile out-of-vehicle travel time (minutes); DIVTT: Transit in-vehicle travel time minus automobile in-vehicle travel time; DCOST: Transit fare minus automobile travel cost (\$).

^b Coefficient equal to 1 by scale normalization

score estimates of the model's parameters. Column 3 shows the half-widths of nominal 90% symmetrical confidence intervals based on the asymptotic normal approximation (half width equals 1.67 times the standard error of the estimate). Column 4 shows half-widths obtained from the bootstrap. The bootstrap confidence intervals are 2.5–3 times wider than the intervals based on the asymptotic normal approximation. The bootstrap confidence interval for the coefficient of DOVTT contains 0, but the confidence interval based on the asymptotic normal approximation does not. Therefore, the hypothesis that the coefficient of DOVTT is zero is not rejected at the 0.1 level based on the bootstrap but is rejected based on the asymptotic normal approximation.

Acknowledgements. Research supported in part by NSF Grant SES-9910925.

References

- Amemiya, T (1985) *Advanced Econometrics*. Harvard University Press, Cambridge.
- Amemiya, T. and Powell, J.L. (1981). A Comparison of the Box-Cox Maximum Likelihood Estimator and the Non-Linear Two-Stage Least Squares Estimator, *Journal of Econometrics*, 17:351–381.
- Cavanagh, C.L. (1987). Limiting Behavior of Estimators Defined by Optimization, unpublished manuscript.
- Chamberlain, G. (1985). Heterogeneity, Omitted Variable Bias, and Duration Dependence. In Heckman, J.J. and Singer, B. (eds), *Longitudinal Analysis of Labor Market Data*, Cambridge University Press, Cambridge, 3–38.

- Chen, S. (2002). Rank Estimation of Transformation Models, *Econometrica*, 70:1683–1697.
- Cheng, S.C., Wei, L.J. and Ying, Z. (1995). Analysis of transformation models with censored data, *Biometrika*, 82:835–845.
- Cheng, S.C., Wei, L.J. and Ying, Z. (1997). Predicting survival probabilities with semiparametric transformation models, *Journal of the American Statistical Association*, 92:227–235.
- Cox, D.R. (1972). Regression Models and Life tables, *Journal of the Royal Statistical Society, Series B*, 34:187–220.
- Delgado, M.A., Rodríguez-Poo, J.M. and Wolf, M. (2001). Subsampling Inference in Cube Root Asymptotics with an Application to Manski's Maximum Score Estimator, *Economics Letters*, 73:241–250.
- Elbers, C. and Ridder, G. (1982). True and Spurious Duration Dependence: The Identifiability of the Proportional Hazard Model, *Review of Economic Studies*, 49:403–409.
- Engle, R.F., Granger, C.W.J., Rice, J. and Weiss, A. (1986). Semiparametric estimates of the relationship between weather and electricity sales. *Journal of the American Statistical Association*, 81:310–320.
- Fan, J. and Gijbels, I. (1996). *Local Polynomial Modelling and Its Applications*. Chapman & Hall, London.
- Foster, A.M., Tian, L. and Wei, L.J. (2001). Estimation for the Box-Cox Transformation Model without Assuming Parametric Error Distribution, *Journal of the American Statistical Association*, 96:1097–1101.
- Gørgens, T. and Horowitz, J.L. (1999). Semiparametric Estimation of a Censored Regression Model with an Unknown Transformation of the Dependent Variable, *Journal of Econometrics*, 90:155–191.
- Härdle, W. (1990) *Applied Nonparametric Regression*. Cambridge University Press, Cambridge.
- Hastie, T.J. and Tibshirani, R.J. (1990). *Generalized Additive Models*. Chapman and Hall, London.
- Heckman, J. and Singer, B. (1984a). A Method for Minimizing the Impact of Distributional Assumptions in Econometric Models for Duration Data, *Econometrica*, 52:271–320.
- Heckman, J. and Singer, B. (1984a). The Identifiability of the Proportional Hazard Model, *Review of Economic Studies*, 51:231–243.
- Honoré, B.E. (1993). Identification Results for Duration Models with Multiple Spells, *Review of Economic Studies*, 60:241–246.
- Horowitz, J.L. (1992). A Smoothed Maximum Score Estimator for the Binary Response Model, *Econometrica*, 60:505–531.
- Horowitz, J.L. (1993a). Semiparametric and Nonparametric Estimation of Quantal Response Models. In Maddala, G.S., Rao, C.R., and Vinod H.D. (eds), *Handbook of Statistics, Vol. 11*, Elsevier, Amsterdam, 45–72.
- Horowitz, J.L. (1993b). Optimal Rates of Convergence of Parameter Estimators in the Binary Response Model with Weak Distributional Assumptions, *Econometric Theory*, 9:1–18.

- Horowitz, J.L. (1993c). Semiparametric Estimation of a Work-Trip Mode Choice Model, *Journal of Econometrics*, 58:49–70.
- Horowitz, J.L. (1996). Semiparametric Estimation of a Regression Model with an Unknown Transformation of the Dependent Variable, *Econometrica*, 64:103–137.
- Horowitz, J.L. (1998). *Semiparametric Methods in Econometrics*. Springer-Verlag, New York.
- Horowitz, J.L. (1999). Semiparametric Estimation of a Proportional Hazard Model with Unobserved Heterogeneity, *Econometrica*, 67:1001–1028.
- Horowitz, J.L. (2001). Nonparametric Estimation of a Generalized Additive Model with an Unknown Link Function. *Econometrica*, 69:499–513.
- Horowitz, J.L. (2002). Bootstrap Critical Values for Tests Based on the Smoothed Maximum Score Estimator, *Journal of Econometrics*, 111:141–167.
- Horowitz, J.L. and Härdle, W. (1996). Direct Semiparametric Estimation of Single-Index Models with Discrete Covariates, *Journal of the American Statistical Association*, 91:1632–1640.
- Horowitz, J.L. and Lee, S. (2004). Semiparametric Estimation of a Panel Data Proportional Hazards Model with Fixed Effects, *Journal of Econometrics*, 119:155–198.
- Hristache, M., Juditsky, A., Polzehl, J. and Spokoiny, V. (2001). Structure Adaptive Approach for Dimension Reduction, *Annals of Statistics*, 29:1537–1566.
- Hristache, M., Juditsky, A. and Spokoiny, V. (2001). Structure Adaptive Approach for Dimension Reduction, *Annals of Statistics*, 29:1–32.
- Ichimura, H. (1993). Semiparametric Least Squares (SLS) and Weighted SLS Estimation of Single-Index Models, *Journal of Econometrics*, 58:71–120.
- Ichimura, H. and Lee, L.-F. (1991). Semiparametric Least Squares Estimation of Multiple Index Models: Single Equation Estimation. In Barnett, W.A., Powell, J., and Tauchen G. (eds), *Nonparametric and Semiparametric Methods in Econometrics and Statistics*. Cambridge University Press, Cambridge, 3–49.
- Ishwaran, H. (1996). Identifiability and Rates of Estimation for Scale Parameters in Location Mixture Models, *Annals of Statistics*, 24:1560–1571.
- Kalbfleisch and Prentice (1980). *The Statistical Analysis of Failure Time Data*, Wiley, New York.
- Kim, J. and Pollard, D. (1990). Cube Root Asymptotics, *Annals of Statistics*, 15:541–551.
- Klein, R.W. and Spady, R.H. (1993). An efficient semiparametric estimator for binary response models. *Econometrica*, 61:387–421.
- Lancaster, T. (1979). Econometric Methods for the Duration of Unemployment, *Econometrica*, 47:939–956.
- Lancaster, T. (2000). The Incidental Parameter Problem Since 1948, *Journal of Econometrics*, 95:391–413.
- Linton, O.B. (1997). Efficient Estimation of Additive Nonparametric Regression Models, *Biometrika*, 84:469–473.
- Linton, O.B. and Härdle, W. (1996). Estimating Additive Regression Models with Known Links, *Biometrika*, 83:529–540.

- Linton, O.B. and Nielsen J.P. (1995). A Kernel Method of Estimating Structured Nonparametric Regression Based on Marginal Integration, *Biometrika*, 82:93–100.
- Mammen, E., Linton, O.B. and Nielsen, J.P. (1999). The Existence and Asymptotic Properties of Backfitting Projection Algorithm under Weak Conditions, *Annals of Statistics*, 27:1443–1490.
- Manski, C.F. (1975). Maximum Score Estimation of the Stochastic Utility Model of Choice, *Journal of Econometrics*, 3:205–228.
- Manski, C.F. (1985). Semiparametric Analysis of Discrete Response: Asymptotic Properties of the Maximum Score Estimator. *Journal of Econometrics*, 27:313–333.
- Manski, C.F. (1988). Identification of Binary Response Models, *Journal of the American Statistical Association*, 83:729–738.
- Matzkin, R.L. (1994). Restrictions of Economic Theory in Nonparametric Methods. In Engle, R.F. and McFadden, D.L. (eds), *Handbook of Econometrics*, Vol. 4. North-Holland, Amsterdam, 2523–2558.
- Meyer, B.D. (1990). Unemployment Insurance and Unemployment Spells, *Econometrica*, 58:757–782.
- Murphy, S.A. (1994). Consistency in a Proportional Hazards Model Incorporating a Random Effect, *Annals of Statistics*, 22:712–731.
- Murphy, S.A. (1995). Asymptotic Theory for the Frailty Model, *Annals of Statistics*, 23:182–198.
- Nielsen, G.G., Gill, R.D., Andersen, P.K. and Sørensen, T.I.A. (1992). A Counting Process Approach to Maximum Likelihood Estimation in Frailty Models, *Scandinavian Journal of Statistics*, 19:25–43, 1992.
- Powell, J.L. (1994). Estimation of Semiparametric Models. In Engle, R.F. and McFadden, D.L. (eds), *Handbook of Econometrics*, Vol. 4. North-Holland, Amsterdam, 2444–2521.
- Powell, J.L., Stock, J.H. and Stoker, T.M. (1989). Semiparametric Estimation of Index Coefficients, *Econometrica*, 51:1403–1430.
- Ridder, G. and Tunalı, I. (1999). Stratified Partial Likelihood Estimation, *Journal of Econometrics*, 92:193–232.
- Robinson, P.M. (1988). Root- N -Consistent Semiparametric Regression. *Econometrica*, 56:931–954.
- Stock, J.H. (1989). Nonparametric Policy Analysis, *Journal of the American Statistical Association*, 84:567–575.
- Stock, J.H. (1991). Nonparametric Policy Analysis: An Application to Estimating Hazardous Waste Cleanup Benefits. In Barnett, W.A., Powell, J., and Tauchen, G. (eds), *Nonparametric and Semiparametric Methods in Econometrics and Statistics*. Cambridge University Press, Cambridge, 77–98.
- Tsiatis, A.A. (1981). A Large Sample Study of Cox's Regression Model, *Annals of Statistics*, 9:93–108.
- Ye, J. and Duan, N. (1997). Nonparametric $n^{-1/2}$ -Consistent Estimation for the General Transformation Model, *Annals of Statistics*, 25:2682–2717.

Bayesian Computational Methods

III.11

Christian P. Robert

11.1	<i>Introduction</i>	721
11.2	<i>Bayesian Computational Challenges</i>	721
	Bayesian Point Estimation	722
	Testing Hypotheses	725
	Model Choice	727
11.3	<i>Monte Carlo Methods</i>	730
	Preamble: Monte Carlo Importance Sampling	730
	First Illustrations	732
	Approximations of the Bayes Factor	738
11.4	<i>Markov Chain Monte Carlo Methods</i>	741
	Metropolis–Hastings as Universal Simulator	741
	Gibbs Sampling and Latent Variable Models	744
	Reversible Jump Algorithms for Variable Dimension Models	748
11.5	<i>More Monte Carlo Methods</i>	753
	Adaptivity for MCMC Algorithms	753
	Population Monte Carlo	758
11.6	<i>Conclusion</i>	763

11.1

Introduction

If, in the mid 1980s, one had asked the average statistician about the difficulties of using Bayesian Statistics, his/her most likely answer would have been “Well, there is this problem of selecting a prior distribution and then, even if one agrees on the prior, the whole Bayesian inference is simply impossible to implement in practice!” The same question asked in the 21st century does not produce the same reply, but rather a much less serious complaint about the lack of generic software (besides winBUGS)! The last 15 years have indeed seen a tremendous change in the way Bayesian Statistics are perceived, both by mathematical statisticians and by applied statisticians and the impetus behind this change has been a prodigious leap-forward in the computational abilities. The availability of very powerful approximation methods has correlatively freed Bayesian modelling, in terms of both model scope *and* prior modelling. As discussed below, a most successful illustration of this gained freedom can be seen in Bayesian model choice, which was only emerging at the beginning of the MCMC era, for lack of appropriate computational tools.

In this chapter, we will first present the most standard computational challenges met in Bayesian Statistics (Sect. 11.2), and then relate these problems with computational solutions. Of course, this chapter is only a terse introduction to the problems and solutions related to Bayesian computations. For more complete references, see Robert and Casella (1999,2004) and Liu (2001), among others. We also restrain from providing an introduction to Bayesian Statistics per se and for comprehensive coverage, address the reader to Robert (2001), (again) among others.

11.2

Bayesian Computational Challenges

Bayesian Statistics being a complete inferential methodology, its scope encompasses the whole range of standard statistician inference (and design), from point estimation to testing, to model selection, and to non-parametrics. In principle, once a prior distribution has been chosen on the proper space, the whole inferential machinery is set and the computation of estimators is usually automatically derived from this setup. Obviously, the practical or numerical derivation of these procedures may be exceedingly difficult or even impossible, as we will see in a few selected examples. Before, we proceed with an incomplete typology of the categories and difficulties met by Bayesian inference. First, let us point out that computational difficulties may originate from one or several of the following items:

- (1) use of a complex parameter space, as for instance in constrained parameter sets like those resulting from imposing stationarity constraints in dynamic models;
- (2) use of a complex sampling model with an intractable likelihood, as for instance in missing data and graphical models;
- (3) use of a huge dataset;

- (4) use of a complex prior distribution (which may be the posterior distribution associated with an earlier sample);
- (5) use of a complex inferential procedure.

Bayesian Point Estimation

11.2.1

In a formalised representation of Bayesian inference, the statistician is given (or she selects) a triplet

- a sampling distribution, $f(x|\theta)$, usually associated with an observation (or a sample) x ;
- a prior distribution $\pi(\theta)$, defined on the parameter space Θ ;
- a loss function $L(\theta, d)$ that compares the decisions (or estimations) d for the true value θ of the parameter.

Using (f, π, L) and an observation x , the Bayesian inference is *always* given as the solution to the minimisation programme

$$\min_d \int_{\Theta} L(\theta, d) f(x|\theta) \pi(\theta) d\theta ,$$

equivalent to the minimisation programme

$$\min_d \int_{\Theta} L(\theta, d) \pi(\theta|x) d\theta .$$

The corresponding procedure is thus associated, for every x , to the solution of the above programme (see, e.g. Robert, 2001, Chap. 2).

There are therefore two levels of computational difficulties with this resolution: first the above integral must be computed. Second, it must be minimised in d . For the most standard losses, like the squared error loss,

$$L(\theta, d) = |\theta - d|^2 ,$$

the solution to the minimisation problem is universally¹ known. For instance, for the squared error loss, it is the posterior mean,

$$\int_{\Theta} \theta \pi(\theta|x) d\theta = \int_{\Theta} \theta f(x|\theta) \pi(\theta) d\theta / \int_{\Theta} f(x|\theta) \pi(\theta) d\theta ,$$

which still requires the computation of both integrals and thus whose complexity depends on the complexity of both $f(x|\theta)$ and $\pi(\theta)$.

¹ In this chapter, the denomination *universal* is used in the sense of *uniformly over all distributions*.

- 1 **Example 1** For a normal distribution $\mathcal{N}(\theta, 1)$, the use of a so-called conjugate prior (see, e.g., Robert, 2001, Chap. 3)

$$\theta \sim \mathcal{N}(\mu, \varepsilon),$$

leads to a closed form expression for the mean, since

$$\begin{aligned} & \int_{\Theta} \theta f(x|\theta) \pi(\theta) \, d\theta \Big/ \int_{\Theta} f(x|\theta) \pi(\theta) \, d\theta = \\ & \int_{\mathbb{R}} \theta \exp \frac{1}{2} \{-\theta^2 (1 + \varepsilon^{-2}) + 2\theta (x + \varepsilon^{-2}\mu)\} \, d\theta \\ & \Big/ \int_{\mathbb{R}} \exp \frac{1}{2} \{-\theta^2 (1 + \varepsilon^{-2}) + 2\theta (x + \varepsilon^{-2}\mu)\} \, d\theta = \frac{x + \varepsilon^{-2}\mu}{1 + \varepsilon^{-2}}. \end{aligned}$$

On the other hand, if we use instead a more involved prior distribution like a poly- t distribution (Bauwens and Richard, 1985),

$$\pi(\theta) = \prod_{i=1}^k [\alpha_i + (\theta - \beta_i)^2]^{-\nu_i} \quad \alpha, \nu > 0$$

the above integrals cannot be computed in closed form anymore. This is *not* a toy example in that the problem may occur after a sequence of t observations, or with a sequence of normal observations whose variance is unknown.

The above example is one-dimensional, but, obviously, bigger challenges await the Bayesian statistician when she wants to tackle high-dimensional problems.

- 2 **Example 2** In a generalised linear model, a conditional distribution of $y \in \mathbb{R}$ given $x \in \mathbb{R}^p$ is defined via a density from an exponential family

$$y|x \sim \exp \{y \cdot \theta(x) - \psi(\theta(x))\}$$

whose natural parameter $\theta(x)$ depends on the conditioning variable x ,

$$\theta(x) = g(\beta^T x), \quad \beta \in \mathbb{R}^p$$

that is, linearly modulo the transform g . Obviously, in practical applications like Econometrics, p can be quite large. Inference on β (which is the true parameter of the model) proceeds through the posterior distribution (where $\mathbf{x} = (x_1, \dots, x_T)$ and $\mathbf{y} = (y_1, \dots, y_T)$)

$$\begin{aligned} \pi(\beta|\mathbf{x}, \mathbf{y}) & \propto \prod_{t=1}^T \exp \{y_t \cdot \theta(x_t) - \psi(\theta(x_t))\} \pi(\beta) \\ & = \exp \left\{ \sum_{t=1}^T y_t \cdot \theta(x_t) - \sum_{t=1}^T \psi(\theta(x_t)) \right\} \pi(\beta), \end{aligned}$$

which rarely is available in closed form. In addition, in some cases ψ may be costly simply to compute and in others T may be large or even very large. Take for instance the case of the dataset processed by Abowd et al. (1999), which covers twenty years of employment histories for over a million workers, with x including indicator variables for over one hundred thousand companies.

A related, although conceptually different, inferential issue concentrates upon *prediction*, that is, the approximation of a distribution related with the parameter of interest, say $g(y|\theta)$, based on the observation of $x \sim f(x|\theta)$. The *predictive distribution* is then defined as

$$\pi(y|x) = \int_{\Theta} g(y|\theta)\pi(\theta|x) d\theta .$$

A first difference with the standard point estimation perspective is obviously that the parameter θ vanishes through the integration. A second and more profound difference is that this parameter is not necessarily well-defined anymore. As will become clearer in a following section, this is a paramount feature in setups where the model is not well-defined and where the statistician hesitates between several (or even an infinity of) models. It is also a case where the standard notion of identifiability is irrelevant, which paradoxically is a “plus” from the computational point of view, as seen below in, e.g., Example 14.

Example 3 Recall that an $AR(p)$ model is given as the *auto-regressive* representation of a time series,

3

$$x_t = \sum_{i=1}^p \theta_i x_{t-i} + \sigma \epsilon_t .$$

It is often the case that the order p of the AR model is not fixed *a priori*, but has to be determined from the data itself. Several models are then competing for the “best” fit of the data, but if the prediction of the next value x_{t+1} is the most important part of the inference, the order p chosen for the best fit is not really relevant. Therefore, all models can be considered in parallel and aggregated through the predictive distribution

$$\pi(x_{t+1}|x_t, \dots, x_1) \propto \int f(x_{t+1}|x_t, \dots, x_{t-p+1})\pi(\theta, p|x_t, \dots, x_1) dp d\theta ,$$

which thus amounts to integrating over the parameters of all models, simultaneously:

$$\sum_{p=0}^{\infty} \int f(x_{t+1}|x_t, \dots, x_{t-p+1}) \pi(\theta|p, x_t, \dots, x_1) d\theta \pi(p|x_t, \dots, x_1) .$$

Note the multiple layers of complexity in this case:

- (1) if the prior distribution on p has an infinite support, the integral simultaneously considers an infinity of models, with parameters of unbounded dimensions;
- (2) the parameter θ varies from model $AR(p)$ to model $AR(p + 1)$, so must be evaluated differently from one model to another. In particular, if the stationarity constraint usually imposed in these models is taken into account, the constraint on $(\theta_1, \dots, \theta_p)$ varies² between model $AR(p)$ and model $AR(p + 1)$;
- (3) prediction is usually used sequentially: every tick/second/hour/day, the next value is predicted based on the past values (x_t, \dots, x_1) . Therefore when t moves to $t + 1$, the entire posterior distribution $\pi(\theta, p | x_t, \dots, x_1)$ must be re-evaluated again, possibly with a very tight time constraint as for instance in financial or radar applications.

We will discuss this important problem in deeper details after the testing section, as part of the model selection problematic.

11.2.2 Testing Hypotheses

A domain where both the philosophy and the implementation of Bayesian inference are at complete odds with the classical approach is the area of testing of hypotheses. At a primary level, this is obvious when opposing the Bayesian evaluation of an hypothesis $H_0 : \theta \in \Theta_0$

$$\Pr^\pi(\theta \in \Theta_0 | x)$$

with a Neyman–Pearson p -value

$$\sup_{\theta \in \Theta_0} \Pr_\theta(T(X) \geq T(x)) ,$$

where T is an appropriate statistic, with observed value $T(x)$. The first quantity involves an integral over the *parameter* space, while the second provides an evaluation over the *observational* space. At a secondary level, the two answers may also strongly disagree even when the number of observations goes to infinity, although there exist cases and priors for which they agree to the order $O(n^{-1})$ or even $O(n^{-3/2})$. (See Robert, 2001, Sect. 3.5.5 and Chap. 5, for more details.)

From a computational point of view, most Bayesian evaluations involve marginal distributions

$$\int_{\Theta_i} f(x | \theta_i) \pi_i(\theta_i) d\theta_i , \quad (11.1)$$

² To impose the stationarity constraint when the order of the $AR(p)$ model varies, it is necessary to reparameterise this model in terms of either the partial autocorrelations or of the roots of the associated lag polynomial. (See, e.g., Robert, 2001, Sect. 4.5.)

where Θ_i and π_i denote the parameter space and the corresponding prior, respectively, under hypothesis H_i ($i = 0, 1$). For instance, the *Bayes factor* is defined as the ratio of the posterior probabilities of the null and the alternative hypotheses over the ratio of the prior probabilities of the null and the alternative hypotheses, i.e.,

$$B_{01}^\pi(x) = \frac{P(\theta \in \Theta_0 | x)}{P(\theta \in \Theta_1 | x)} \bigg/ \frac{\pi(\theta \in \Theta_0)}{\pi(\theta \in \Theta_1)} .$$

This quantity is instrumental in the computation of the posterior probability

$$P(\theta \in \Theta_0 | x) = \frac{1}{1 + B_{10}^\pi(x)}$$

under equal prior probabilities for both Θ_0 and Θ_1 . It is also the central tool in practical (as opposed to decisional) Bayesian testing (Jeffreys, 1961) as the Bayesian equivalent of the likelihood ratio.

The first ratio in $B_{01}^\pi(x)$ is then the ratio of integrals of the form (11.1) and it is rather common to face difficulties in the computation of *both* integrals.³

Example 4: *Continuation of Example 2* In the case of the generalised linear model, a standard testing situation is to decide whether or not a factor, x_1 say, is influential on the dependent variable y . This is often translated as testing whether or not the corresponding component of β , β_1 , is *equal* to 0, i.e. $\Theta_0 = \{\beta; \beta_1 = 0\}$. If we denote by β_{-1} the *other* components of β , the Bayes factor for this hypothesis will be

4

$$\int_{\mathbb{R}^p} \exp \left\{ \sum_{t=1}^T y_t \cdot g(\beta^T x_t) - \sum_{t=1}^T \psi(g(\beta^T x_t)) \right\} \pi(\beta) \, d\beta \bigg/ \int_{\mathbb{R}^{p-1}} \exp \left\{ \sum_{t=1}^T y_t \cdot g(\beta_{-1}^T(x_t)_{-1}) - \sum_{t=1}^T \psi(\beta_{-1}^T(x_t)_{-1}) \right\} \pi_{-1}(\beta_{-1}) \, d\beta_{-1} ,$$

when π_{-1} is the prior constructed for the null hypothesis and when the prior weights of H_0 and of the alternative are both equal to $1/2$. Obviously, besides the normal conjugate case, both integrals cannot be computed in a closed form.

In a related manner, *confidence regions* are also mostly intractable, being defined through the solution to an implicit equation. Indeed, the Bayesian confidence region for a parameter θ is defined as the *highest posterior region*,

$$\{\theta; \pi(\theta|x) \geq k(x)\} , \tag{11.2}$$

³ In this presentation of Bayes factors, we completely bypass the methodological difficulty of defining $\pi(\theta \in \Theta_0)$ when Θ_0 is of measure 0 for the original prior π and refer the reader to Robert (2001, Section 5.2.3) for proper coverage of this issue.

where $k(x)$ is determined by the coverage constraint

$$\Pr^\pi(\pi(\theta|x) \geq k(x)|x) = \alpha ,$$

α being the confidence level. While the normalising constant is not necessary to construct a confidence region, the resolution of the implicit equation (11.2) is rarely straightforward!

5 Example 5 Consider a binomial observation $x \sim \mathcal{B}(n, \theta)$ with a conjugate prior distribution, $\theta \sim \mathcal{Be}(\gamma_1, \gamma_2)$. In this case, the posterior distribution is available in closed form,

$$\theta|x \sim \mathcal{Be}(\gamma_1 + x, \gamma_2 + n - x) .$$

However, the determination of the θ 's such that

$$\theta^{\gamma_1+x-1}(1-\theta)^{\gamma_2+n-x-1} \geq k(x)$$

with

$$\Pr^\pi (\theta^{\gamma_1+x-1}(1-\theta)^{\gamma_2+n-x-1} \geq k(x)|x) = \alpha$$

is not possible analytically. It actually implies two levels of numerical difficulties:

Step 1 find the solution(s) to $\theta^{\gamma_1+x-1}(1-\theta)^{\gamma_2+n-x-1} = k$,

Step 2 find the k corresponding to the right coverage,

and each value of k examined in Step 2. requires a new resolution of Step 1.

The setting is usually much more complex when θ is a multidimensional parameter, because the interest is usually in getting marginal confidence sets. Example 2 is an illustration of this setting: deriving a confidence region on one component, β_1 say, first involves computing the marginal posterior distribution of this component. As in Example 4, the integral

$$\int_{\mathbb{R}^{p-1}} \exp \left\{ \sum_{t=1}^T y_t \cdot g(\beta^T x_t) - \sum_{t=1}^T \psi(\beta^T x_t) \right\} \pi_{-1}(\beta_{-1}) \, d\beta_{-1} ,$$

which is proportional to $\pi(\beta_1|x)$, is most often intractable.

11.2.3 Model Choice

We distinguish *model choice* from testing, not only because it leads to further computational difficulties, but also because it encompasses a larger scope of inferential goals than mere testing. Note first that model choice has been the subject of

considerable effort in the past years, and has seen many advances, including the coverage of problems of higher complexity and the introduction of new concepts. We stress that such advances mostly owe to the introduction of new computational methods.

As discussed in further details in Robert (2001, Chap. 7), the inferential action related with model choice does take place on a wider scale: it covers and compares models, rather than parameters, which makes the sampling distribution $f(x)$ “more unknown” than simply depending on an undetermined parameter. In some respect, it is thus closer to estimation than to regular testing. In any case, it requires a more precise evaluation of the consequences of choosing the “wrong” model or, equivalently of deciding which model is the most appropriate to the data at hand. It is thus both broader and less definitive as deciding whether $H_0 : \theta_1 = 0$ is true. At last, the larger inferential scope mentioned in the first point means that we are leaving for a while the well-charted domain of solid parametric models.

From a computational point of view, model choice involves more complex structures that, almost systematically, require advanced tools, like simulation methods which can handle collections of parameter spaces (also called *spaces of varying dimensions*), specially designed for model comparison.

Example 6 A mixture of distributions is the representation of a distribution (density) as the weighted sum of standard distributions (densities). For instance, a mixture of Poisson distributions, denoted as

6

$$\sum_{i=1}^k p_i \mathcal{P}(\lambda_i)$$

has the following density:

$$\Pr(X = k) = \sum_{i=1}^k p_i \frac{\lambda_i^k}{k!} e^{-\lambda_i} .$$

This representation of distributions is multi-faceted and can be used in populations with known heterogeneities (in which case a component of the mixture corresponds to an homogeneous part of the population) as well as a non-parametric modelling of unknown populations. This means that, in some cases, k is known and, in others, it is both unknown and part of the inferential problem.

First, consider the setting where several (parametric) models are in competition,

$$\mathfrak{M}_i : x \sim f_i(x|\theta_i) , \quad \theta_i \in \Theta_i , \quad i \in I ,$$

the index set I being possibly infinite. From a Bayesian point of view, a prior distribution must be constructed for each model \mathfrak{M}_i as if it were the only and true model under consideration since, in most perspectives except *model averaging*,

one of these models will be selected and used as the only and true model. The parameter space associated with the above set of models can be written as

$$\Theta = \bigcup_{i \in I} \{i\} \times \Theta_i, \quad (11.3)$$

the model indicator $\mu \in I$ being now part of the parameters. So, if the statistician allocates probabilities p_i to the indicator values, that is, to the models \mathfrak{M}_i ($i \in I$), and if she then defines priors $\pi_i(\theta_i)$ on the parameter subspaces Θ_i , things fold over by virtue of Bayes's theorem, as usual, since she can compute

$$p(\mathfrak{M}_i|x) = P(\mu = i|x) = \frac{p_i \int_{\Theta_i} f_i(x|\theta_i) \pi_i(\theta_i) d\theta_i}{\sum_j p_j \int_{\Theta_j} f_j(x|\theta_j) \pi_j(\theta_j) d\theta_j}.$$

While a common solution based on this prior modeling is simply to take the (marginal) MAP estimator of μ , that is, to determine the model with the largest $p(\mathfrak{M}_i|x)$, or even to use directly the average

$$\sum_j p_j \int_{\Theta_j} f_j(y|\theta_j) \pi_j(\theta_j|x) d\theta_j = \sum_j p(\mathfrak{M}_j|x) m_j(y)$$

as a predictive density in y in *model averaging*, a deeper-decision theoretic evaluation is often necessary.

7 **Example 7:** (*Continuation of Example 3*) In the setting of the $AR(p)$ models, when the order p of the dependence is unknown, model averaging as presented in Example 3 is not always a relevant solution when the statistician wants to estimate this order p for different purposes. Estimation is then a more appropriate perspective than testing, even though care must be taken because of the discrete nature of p . (For instance, the posterior expectation of p is not an appropriate estimator!)

8 **Example 8** Spiegelhalter et al. (2002) have developed a Bayesian approach to model choice that appears like an alternative to both Akaike's and Schwartz Information Criterion, called DIC (for Deviance Information Criterion). For a model with density $f(x|\theta)$ and a prior distribution $\pi(\theta)$, the deviance is defined as $D(\theta) = -2 \log(f(x|\theta))$ but this is not a good discriminating measure between models because of its bias toward higher dimensional models. The penalized deviance of Spiegelhalter et al. (2002) is

$$\text{DIC} = \mathbb{E}[D(\theta)|x] + \{\mathbb{E}[D(\theta)|x] - D(\mathbb{E}[\theta|x])\},$$

with the “best” model associated with the smallest DIC. Obviously, the computation of the posterior expectation $\mathbb{E}[D(\theta)|x] = -2\mathbb{E}[\log(f(x|\theta))|x]$ is complex outside exponential families.

As stressed earlier in this section, the computation of predictive densities, marginals, Bayes factors, and other quantities related to the model choice procedures is generally very involved, with specificities that call for tailor-made solutions:

- The computation of integrals is increased by a factor corresponding to the number of models under consideration.
- Some parameter spaces are infinite-dimensional, as in non-parametric settings and that may cause measure-theoretic complications.
- The computation of posterior or predictive quantities involves integration over different parameter spaces and thus increases the computational burden, since there is no time savings from one subspace to the next.
- In some settings, the size of the collection of models is very large or even infinite and some models cannot be explored. For instance, in Example 4, the collection of all submodels is of size 2^p and some pruning method must be found in variable selection to avoid exploring the whole tree of all submodels.

Monte Carlo Methods

11.3

The natural approach to these computational problems is to use computer simulation and Monte Carlo techniques, rather than numerical methods, simply because there is much more to gain from exploiting the probabilistic properties of the integrands rather than their analytical properties. In addition, the dimension of most problems considered in current Bayesian Statistics is such that very involved numerical methods should be used to provide a satisfactory approximation in such integration or optimisation problems. Indeed, down-the-shelf numerical methods cannot handle integrals in dimensions larger than 4 and more advanced numerical integration methods require analytical studies on the distribution of interest.

Preamble: Monte Carlo Importance Sampling

11.3.1

Given the statistical nature of the problem, the approximation of an integral like

$$\mathfrak{J} = \int_{\Theta} h(\theta)f(x|\theta)\pi(\theta) \, d\theta ,$$

should indeed take advantage of the special nature of \mathfrak{J} , namely, the fact that π is a probability density⁴ or, instead, that $f(x|\theta)\pi(\theta)$ is proportional to a density. As de-

⁴ The prior distribution can be used for importance sampling only if it is a proper prior and not a σ -finite measure.

tailed in Chap. II.2 this volume, or in Robert and Casella (2004, Chap. 3), the *Monte Carlo method* was introduced by Metropolis and Ulam (1949) and Von Neumann (1951) for this purpose. For instance, if it is possible to generate (via a computer) random variables $\theta_1, \dots, \theta_m$ from $\pi(\theta)$, the average

$$\frac{1}{m} \sum_{i=1}^m h(\theta_i) f(x|\theta_i)$$

converges (almost surely) to \mathfrak{J} when m goes to $+\infty$, according to the Law of Large Numbers. Obviously, if an i.i.d. sample of θ_i 's from the posterior distribution $\pi(\theta|x)$ can be produced, the average

$$\frac{1}{m} \sum_{i=1}^m h(\theta_i) \tag{11.4}$$

converges to

$$\mathbb{E}^\pi[h(\theta)|x] = \frac{\int_{\Theta} h(\theta) f(x|\theta) \pi(\theta) \, d\theta}{\int_{\Theta} f(x|\theta) \pi(\theta) \, d\theta}$$

and it usually is more interesting to use this approximation, rather than

$$\frac{\sum_{i=1}^m h(\theta_i) f(x|\theta_i)}{\sum_{i=1}^m f(x|\theta_i)}$$

when the θ_i 's are generated from $\pi(\theta)$, especially when $\pi(\theta)$ is flat compared with $\pi(\theta|x)$.

In addition, if the posterior variance $\text{var}(h(\theta)|x)$ is finite, the Central Limit Theorem applies to the empirical average (11.4), which is then asymptotically normal with variance $\text{var}(h(\theta)|x)/m$. Confidence regions can then be built from this normal approximation and, most importantly, the magnitude of the error remains of order $1/\sqrt{m}$, whatever the dimension of the problem, in opposition with numerical methods.⁵ (See also Robert and Casella, 2004, Chap. 4, for more details on the convergence assessment based on the CLT.)

The Monte Carlo method actually applies in a much wider generality than the above simulation from π . For instance, because \mathfrak{J} can be represented in an infinity of ways as an expectation, there is no need to simulate from the distributions $\pi(\cdot|x)$ or π to get a good approximation of \mathfrak{J} . Indeed, if g is a probability density

⁵ The constant order of the Monte Carlo error does not imply that the computational effort remains the same as the dimension increases, most obviously, but rather that the decrease (with m) in variation has the rate $1/\sqrt{m}$.

with $\text{supp}(g)$ including the support of $|h(\theta)|f(x|\theta)\pi(\theta)$, the integral \mathfrak{J} can also be represented as an expectation against g , namely

$$\int \frac{h(\theta)f(x|\theta)\pi(\theta)}{g(\theta)}g(\theta) d\theta .$$

This representation leads to the *Monte Carlo method with importance function* g : generate $\theta_1, \dots, \theta_m$ according to g and approximate \mathfrak{J} through

$$\frac{1}{m} \sum_{i=1}^m h(\theta_i)\omega_i(\theta_i) ,$$

with the weights $\omega(\theta_i) = f(x|\theta_i)\pi(\theta_i)/g(\theta_i)$. Again, by the Law of Large Numbers, this approximation almost surely converges to \mathfrak{J} . And this estimator is unbiased. In addition, an approximation to $\mathbb{E}^\pi[h(\theta)|x]$ is given by

$$\frac{\sum_{i=1}^m h(\theta_i)\omega(\theta_i)}{\sum_{i=1}^m \omega(\theta_i)} . \tag{11.5}$$

since the numerator and denominator converge to

$$\int_{\Theta} h(\theta)f(x|\theta)\pi(\theta) d\theta \quad \text{and} \quad \int_{\Theta} f(x|\theta)\pi(\theta) d\theta ,$$

respectively, if $\text{supp}(g)$ includes $\text{supp}(f(x|\cdot)\pi)$. Notice that the ratio (11.5) does not depend on the normalizing constants in either $h(\theta)$, $f(x|\theta)$ or $\pi(\theta)$. The approximation (11.5) can therefore be used in settings when some of these normalizing constants are unknown. Notice also that the *same* sample of θ_i 's can be used for the approximation of both the numerator and denominator integrals: even though using an estimator in the denominator creates a bias, (11.5) does converge to $\mathbb{E}^\pi[h(\theta)|x]$.

While this convergence is guaranteed for all densities g with wide enough support, the choice of the importance function is crucial. First, simulation from g must be easily implemented. Moreover, the function $g(\theta)$ must be close enough to the function $h(\theta)\pi(\theta|x)$, in order to reduce the variability of (11.5) as much as possible; otherwise, most of the weights $\omega(\theta_i)$ will be quite small and a few will be overly influential. In fact, if $\mathbb{E}^h[h^2(\theta)\omega^2(\theta)]$ is not finite, the variance of the estimator (11.5) is infinite (see Robert and Casella, 2004, Chap. 3). Obviously, the dependence on g of the importance function h can be avoided by proposing generic choices such as the posterior distribution $\pi(\theta|x)$.

First Illustrations

In either point estimation or simple testing situations, the computational problem is often expressed as a ratio of integrals. Let us start with a toy example to set up the way Monte Carlo methods proceed and highlight the difficulties of applying a generic approach to the problem.

- 9** **Example 9** Consider a t -distribution $\mathcal{T}(\nu, \theta, 1)$ sample (x_1, \dots, x_n) with ν known. Assume in addition a flat prior $\pi(\theta) = 1$ as in a non-informative environment. While the posterior distribution on θ can be easily plotted, up to a normalising constant (Fig. 11.1), because we are in dimension 1, direct simulation and computation from this posterior is impossible.

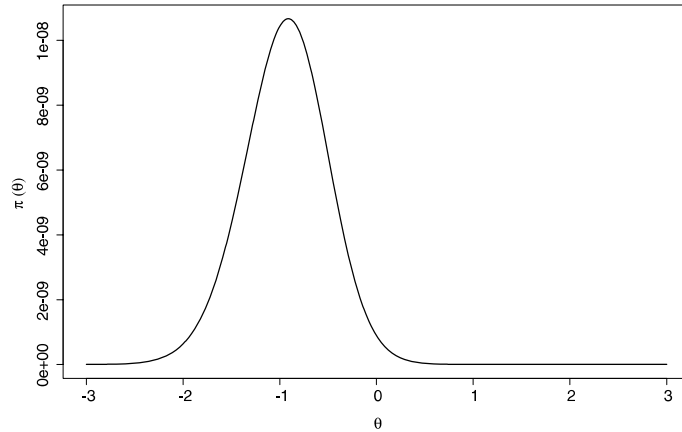


Figure 11.1. Posterior density of θ in the setting of Example 9 for $n = 10$, with a simulated sample from $\mathcal{T}(3, 0, 1)$

If the inferential problem is to decide about the value of θ , the posterior expectation is

$$\mathbb{E}^\pi[\theta | x_1, \dots, x_n] = \frac{\int \theta \prod_{i=1}^n [v + (x_i - \theta)^2]^{-(v+1)/2} d\theta}{\int \prod_{i=1}^n [v + (x_i - \theta)^2]^{-(v+1)/2} d\theta}.$$

This ratio of integrals is not directly computable. Since $(v + (x_i - \theta)^2)^{-(v+1)/2}$ is proportional to a t -distribution $\mathcal{T}(\nu, x_i, 1)$ density, a solution to the approximation of the integrals is to use *one* of the i 's to "be" the density in both integrals. For instance, if we generate $\theta_1, \dots, \theta_m$ from the $\mathcal{T}(\nu, x_1, 1)$ distribution, the equivalent of (11.5) is

$$\delta_m^\pi = \frac{\sum_{j=1}^m \theta_j \prod_{i=2}^n [v + (x_i - \theta_j)^2]^{-(v+1)/2}}{\sum_{j=1}^m \prod_{i=2}^n [v + (x_i - \theta_j)^2]^{-(v+1)/2}} \quad (11.6)$$

since the first term in the product has been “used” for the simulation and the normalisation constants have vanished in the ratio. Figure 11.2 is an illustration of the speed of convergence of this estimator to the true posterior expectation: it provides the evolution of δ_m^π as a function of m both on average and on range (provided by repeated simulations of δ_m^π). As can be seen from the graph, the average is almost constant from the start, as it should, because of unbiasedness, while the range decreases very slowly, as it should, because of extreme value theory. The graph provides in addition the 90% empirical confidence interval built on these simulations.⁶ Both the range and the empirical confidence intervals are decreasing in $1/\sqrt{n}$, as expected from the theory. (This is further established by regressing both the log-ranges and the log-lengths of the confidence intervals on $\log(n)$, with slope equal to -0.5 in both cases, as shown by Fig. 11.3.)

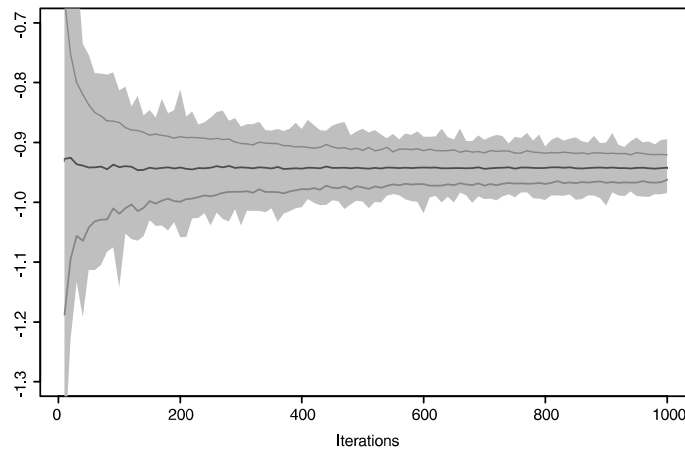


Figure 11.2. Evolution of a sequence of 500 estimators (11.6) over 1000 iterations: range (in gray), 0.05 and 0.95 quantiles, and average, obtained on the same sample as in Fig. 11.1 when simulating from the t distribution with location x_1

Now, there is a clear arbitrariness in the choice of x_1 in the sample (x_1, \dots, x_n) for the proposal $\mathcal{T}(v, x_1, 1)$. While any of the x_i 's has the same theoretical validity to “represent” the integral and the integrating density, the choice of x_i 's closer to the posterior mode (the true value of θ is 0) induces less variability in the estimates, as shown by a further simulation experiment through Fig. 11.4. It is fairly clear from this comparison that the choice of extremal values like $x_{(1)} = -3.21$ and even more $x_{(10)} = 1.72$ is detrimental to the quality of the approximation, compared with the median $x_{(5)} = -0.86$. The range of the estimators is much wider for both

⁶ The empirical (Monte Carlo) confidence interval is not to be confused with the asymptotic confidence interval derived from the normal approximation. As discussed in Robert and Casella (2004, Chap. 4), these two intervals may differ considerably in width, with the interval derived from the CLT being much more optimistic!

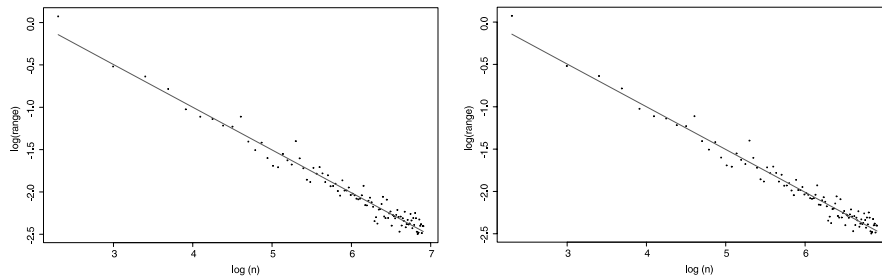


Figure 11.3. Regression of the log-ranges (*left*) and the log-lengths of the confidence intervals (*right*) on $\log(n)$, for the output in Fig. 11.2

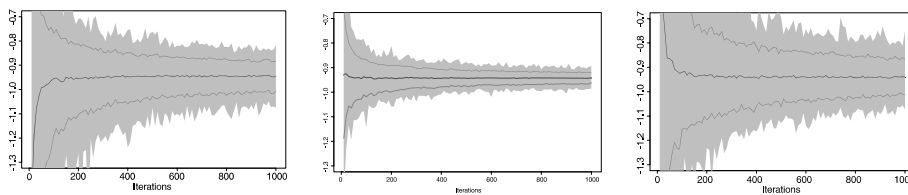


Figure 11.4. Repetition of the experiment described in Fig. 11.2 for three different choices of x_i : $\min x_i$, $x_{(5)}$ and $\max x_i$ (*from left to right*)

extremes, but the influence of this choice is also visible for the average which does not converge so quickly.⁷

This example thus shows that Monte Carlo methods, while widely available, may easily face inefficiency problems when the simulated values are not sufficiently attuned to the distribution of interest. It also shows that, fundamentally, there is no difference between importance sampling and regular Monte Carlo, in that the integral \mathcal{J} can naturally be represented in many ways.

Although we do not wish to spend too much space on this issue, let us note that the choice of the importance function gets paramount when the support of the function of interest is *not* the whole space. For instance, a tail probability, associated with $h(\theta) = \mathbb{1}_{\theta \geq \theta_0}$ say, should be estimated with an importance function whose support is $[\theta_0, \infty)$. (See Robert and Casella, 2004, Chap. 3, for details.)

10

Example 10: (*Continuation of Example 9*) If, instead, we wish to consider the probability that $\theta \geq 0$, using the t -distribution $\mathcal{T}(v, x_i, 1)$ is not a good idea because negative values of θ are somehow simulated “for nothing”. A better proposal (in

⁷ An alternative to the simulation from one $\mathcal{T}(v, x_i, 1)$ distribution that does not require an extensive study on the most appropriate x_i is to use a mixture of the $\mathcal{T}(v, x_i, 1)$ distributions. As seen in Sect. 11.5.2, the weights of this mixture can even be optimised automatically.

terms of variance) is to use the “folded” t -distribution $\mathcal{T}(\nu, x_i, 1)$, with density proportional to

$$\psi_i(\theta) = [\nu + (x_i - \theta)^2]^{-(\nu+1)/2} + [\nu + (x_i + \theta)^2]^{-(\nu+1)/2} ,$$

on \mathbb{R}_+ , which can be simulated by taking the absolute value of a $\mathcal{T}(\nu, x_i, 1)$ rv. All simulated values are then positive and the estimator of the probability is

$$e_m^\pi = \frac{\sum_{j=1}^m \prod_{i \neq k} [\nu + (x_i - |\theta_j|)^2]^{-(\nu+1)/2}}{\sum_{j=1}^m \prod_{i \neq k} [\nu + (x_i - \theta_j)^2]^{-(\nu+1)/2}} \bigg/ \psi_k(|\theta_j|) \quad (11.7)$$

where the θ_j 's are iid $\mathcal{T}(\nu, x_k, 1)$. Note that this is a very special occurrence where the *same* sample can be used in both the numerator and the denominator. In fact, in most cases, two different samples have to be used, if only because the support of the importance distribution for the numerator is *not* the whole space, unless, of course, all normalising constants are known. Figure 11.5 reproduces earlier figures for this problem, when using $x_{(5)}$ as the parameter of the t distribution.

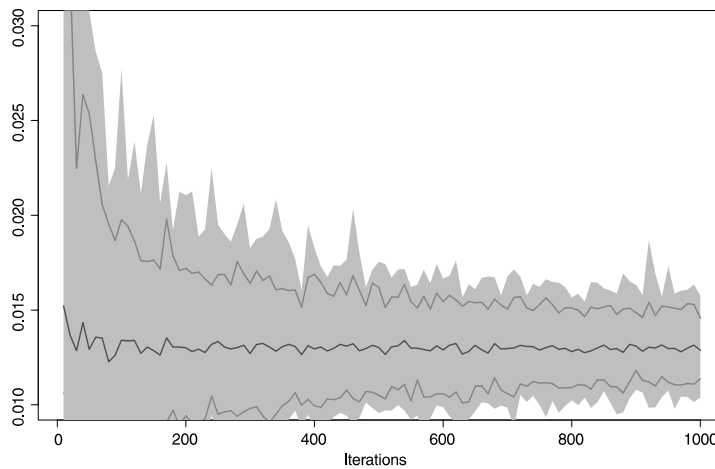


Figure 11.5. Evolution of a sequence of 100 estimators (11.7) over 1000 iterations (same legend as Fig. 11.2)

The above example is one-dimensional (in the parameter) and the problems exhibited there can be found severalfold in multidimensional settings. Indeed, while

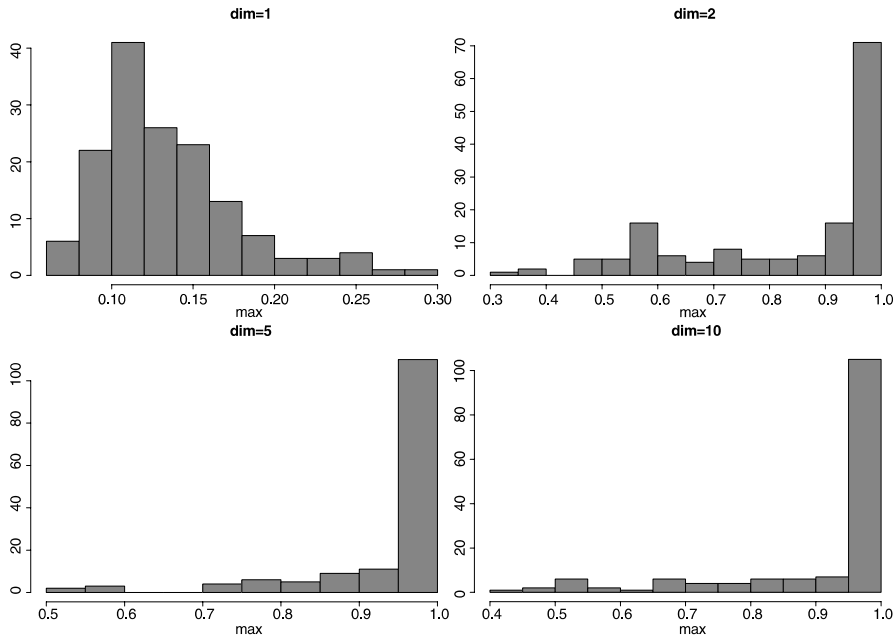


Figure 11.6. Comparison of the distribution of the largest importance weight based upon 150 replications of an importance sampling experiment with 245 observations and dimensions $p = 1, 2, 5, 10$

Monte Carlo methods do not suffer from the “curse of dimension” in the sense that the error of the corresponding estimators is always decreasing in $1/\sqrt{n}$, notwithstanding the dimension, it gets increasingly difficult to come up with satisfactory importance sampling distributions as the dimension gets higher and higher. As we will see in Sect. 11.5, the intuition built on MCMC methods has to be exploited to derive satisfactory importance functions.

11

Example 11: (Continuation of Example 2) A particular case of generalised linear model is the *probit model*,

$$\Pr_{\theta}(Y = 1|x) = 1 - \Pr_{\theta}(Y = 0|x) = \Phi(x^T\theta) \quad \theta \in \mathbb{R}^p,$$

where Φ denotes the normal $\mathcal{N}(0, 1)$ cdf. Under a flat prior $\pi(\theta) = 1$, for a sample $(x_1, y_1), \dots, (x_n, y_n)$, the corresponding posterior distribution is proportional to

$$\prod_{i=1}^n \Phi(x_i^T\theta)^{y_i} \Phi(-x_i^T\theta)^{1-y_i}. \quad (11.8)$$

Direct simulation from this distribution is obviously impossible since the very computation of $\Phi(z)$ is a difficulty in itself. If we pick an importance function

for this problem, the adequation with the posterior distribution will need to be better and better as the dimension p increases. Otherwise, the repartition of the weights will get increasingly asymmetric: very few weights will be different from 0.

Figure 11.6 illustrates this degeneracy of the importance sampling approach as the dimension increases. We simulate parameters β 's and datasets (x_i, y_i) ($i = 1, \dots, 245$) for dimensions p ranging from 1 to 10, then represented the histograms of the largest weight for $p = 1, 2, 5, 10$. The x_i 's were simulated from a $\mathcal{N}_p(0, I_p)$ distribution, while the importance sampling distribution was a $\mathcal{N}_p(0, I_p/p)$ distribution.

Approximations of the Bayes Factor

As explained in Sects. 11.2.2 and 11.2.3, the first computational difficulty associated with Bayesian testing is the derivation of the *Bayes factor*, of the form

$$B_{12}^\pi = \frac{\int_{\Theta_1} f_1(x|\theta_1)\pi_1(\theta_1) d\theta_1}{\int_{\Theta_2} f_2(x|\theta_2)\pi_2(\theta_2) d\theta_2} = \frac{m_1(x)}{m_2(x)},$$

where, for simplicity's sake, we have adopted the model choice perspective (that is, θ_1 and θ_2 may live in completely different spaces).

Specific Monte Carlo methods for the estimation of ratios of normalizing constants, or, equivalently, of Bayes factors, have been developed in the past five years. See Chen et al. (2000, Chap. 5) for a complete exposition. In particular, the importance sampling technique is rather well-adapted to the computation of those Bayes factors: Given an importance distribution, with density proportional to g , and a sample $\theta^{(1)}, \dots, \theta^{(T)}$ simulated from g , the marginal density for model \mathfrak{M}_i , $m_i(x)$, is approximated by

$$\widehat{m}_i(x) = \sum_{t=1}^T f_i(x|\theta^{(t)}) \frac{\pi_i(\theta^{(t)})}{g(\theta^{(t)})} \bigg/ \sum_{t=1}^T \frac{\pi_i(\theta^{(t)})}{g(\theta^{(t)})},$$

where the denominator takes care of the (possibly) missing normalizing constants. (Notice that, if g is a density, the expectation of $\pi(\theta^{(t)})/g(\theta^{(t)})$ is 1 and the denominator should be replaced by T to decrease the variance of the estimator of $m_i(x)$.) A compelling incentive, among others, for using importance sampling in the setting of model choice is that the sample $(\theta^{(1)}, \dots, \theta^{(T)})$ can be recycled for all models \mathfrak{M}_i sharing the same parameters (in the sense that the models \mathfrak{M}_i are parameterized in the same way, e.g. by their first moments).

12 Example 12: (*Continuation of Example 4*) In the case the β 's are simulated from a product instrumental distribution

$$g(\beta) = \prod_{i=1}^P g_i(\beta_i),$$

the sample of β 's produced for the general model of Example 2, \mathfrak{M}_1 say, can also be used for the restricted model, \mathfrak{M}_2 , where $\beta_1 = 0$, simply by deleting the first component and keeping the following components, with the corresponding importance density being

$$g_{-1}(\beta) = \prod_{i=2}^P g_i(\beta_i).$$

Once the β 's have been simulated, the Bayes factor B_{12}^π can be approximated by $\widehat{m}_1(x)/\widehat{m}_2(x)$.

However, the variance of $\widehat{m}(x)$ may be infinite, depending on the choice of g . A possible choice is $g(\theta) = \pi(\theta)$, with wider tails than $\pi(\theta|x)$, but this is often inefficient if the data is informative because the prior and the posterior distributions will be quite different and most of the simulated values $\theta^{(t)}$ fall outside the modal region of the likelihood. For the choice $g(\theta) = f(x|\theta)\pi(\theta)$,

$$\widehat{m}(x) = 1 / \frac{1}{T} \sum_{t=1}^T \frac{1}{f(x|\theta^{(t)})}, \quad (11.9)$$

is the *harmonic mean* of the likelihoods, but the corresponding variance is infinite when the likelihood has thinner tails than the prior (which is often the case). Explicitly oriented towards the computation of ratios of normalising constants, *bridge sampling* was introduced in Meng and Wong (1996): if both models \mathfrak{M}_1 and \mathfrak{M}_2 cover the same parameter space Θ , if $\pi_1(\theta|x) = c_1 \tilde{\pi}_1(\theta|x)$ and $\pi_2(\theta|x) = c_2 \tilde{\pi}_2(\theta|x)$, where c_1 and c_2 are unknown normalising constants, then the equality

$$\frac{c_2}{c_1} = \frac{\mathbb{E}^{\pi_2} [\tilde{\pi}_1(\theta|x) h(\theta)]}{\mathbb{E}^{\pi_1} [\tilde{\pi}_2(\theta|x) h(\theta)]}$$

holds for any *bridge function* $h(\theta)$ such that both expectations are finite. The *bridge sampling estimator* is then

$$B_{12}^S = \frac{\frac{1}{n_1} \sum_{i=1}^{n_1} \tilde{\pi}_2(\theta_{1i}|x) h(\theta_{1i})}{\frac{1}{n_2} \sum_{i=1}^{n_2} \tilde{\pi}_1(\theta_{2i}|x) h(\theta_{2i})},$$

where the θ_{ji} 's are simulated from $\pi_j(\theta|x)$ ($j = 1, 2, i = 1, \dots, n_j$). For instance, if

$$h(\theta) = 1 / [\bar{\pi}_1(\theta|x)\bar{\pi}_2(\theta_{1i}|x)] ,$$

then B_{12}^S is a ratio of harmonic means, generalizing (11.9). Meng and Wong (1996) have derived an (asymptotically) optimal bridge function

$$h^*(\theta) = \frac{n_1 + n_2}{n_1 \pi_1(\theta|x) + n_2 \pi_2(\theta|x)} .$$

This choice is not of direct use, since the normalizing constants of $\pi_1(\theta|x)$ and $\pi_2(\theta|x)$ are unknown (otherwise, we should not need to resort to such techniques!). Nonetheless, it shows that a good bridge function should cover the support of both posteriors, with equal weights if $n_1 = n_2$.

Example 13: (Continuation of Example 2) For generalized linear models, the mean (conditionally on the covariates) satisfies

13

$$\mathbb{E}[y|\theta] = \nabla \psi(\theta) = \Psi(x^t \beta) ,$$

where Ψ is the link function. The choice of the link function Ψ usually is quite open. For instance, when the y 's take values in $\{0, 1\}$, three common choices of Ψ are (McCullagh and Nelder, 1989)

$$\Psi_1(t) = \exp(t)/(1 + \exp(t)) , \quad \Psi_2(t) = \Phi(t) , \quad \text{and} \quad \Psi_3(t) = 1 - \exp(-\exp(t)) ,$$

corresponding to the *logit*, *probit* and *log-log* link functions (where Φ denotes the c.d.f. of the $\mathcal{N}(0, 1)$ distribution). If the prior distribution π on the β 's is a normal $\mathcal{N}_p(\xi, \tau^2 I_p)$, and if the bridge function is $h(\beta) = 1/\pi(\beta)$, the bridge sampling estimate is then ($1 \leq i < j \leq 3$)

$$B_{ij}^S = \frac{\frac{1}{n} \sum_{t=1}^n f_j(\beta_{it}|x)}{\frac{1}{n} \sum_{t=1}^n f_i(\beta_{jt}|x)} ,$$

where the β_{it} are generated from $\pi_i(\beta_i|x) \propto f_i(\beta_i|x)\pi(\beta_i)$, that is, from the true posteriors for each link function.

As can be seen from the previous developments, such methods require a rather careful tuning to be of any use. Therefore, they are rather difficult to employ outside settings where pairs of models are opposed. In other words, they cannot be directly used in general model choice settings where the parameter space (and in particular the parameter dimension) varies across models, like, for instance, Example 7. To address the computational issues corresponding to these cases requires more advanced techniques introduced in the next section.

11.4

Markov Chain Monte Carlo Methods

As described precisely in Chap. III.3 and in Robert and Casella (2004), MCMC methods try to overcome the limitation of regular Monte Carlo methods by mean of a Markov chain with stationary distribution the posterior distribution. There exist rather generic ways of producing such chains, including Metropolis–Hastings and Gibbs algorithms. Besides the fact that stationarity of the target distribution is enough to justify a simulation method by Markov chain generation, the idea at the core of MCMC algorithms is that local exploration, when properly weighted, can lead to a valid representation of the distribution of interest, as for instance, the Metropolis–Hastings algorithm.

11.4.1

Metropolis–Hastings as Universal Simulator

The Metropolis–Hastings, presented in Robert and Casella (2004) and Chap. II.3, offers a straightforward solution to the problem of simulating from the posterior distribution $\pi(\theta|x) \propto f(x|\theta) \pi(\theta)$: starting from an arbitrary point θ_0 , the corresponding Markov chain explores the surface of this posterior distribution by a random walk proposal $q(\theta|\theta')$ that progressively visits the whole range of the possible values of θ .

Metropolis–Hastings Algorithm

At iteration t

1. Generate $\xi \sim q(\xi|\theta^{(t)})$, $u_t \sim \mathcal{U}([0, 1])$
2. Take

$$\theta^{(t+1)} = \begin{cases} \xi & \text{if } u_t \leq \frac{\pi(\xi|x) q(\theta^{(t)}|\xi)}{\pi(\theta^{(t)}|x) q(\xi|\theta^{(t)})} \\ \theta^{(t)} & \text{otherwise} \end{cases}$$

14

Example 14: (Continuation of Example 11) In the case $p = 1$, the probit model defined in Example 11 can also be over-parameterised as

$$P(Y_i = 1|x_i) = 1 - P(Y_i = 0|x_i) = \Phi(x_i\beta/\sigma),$$

since it only depends on β/σ . The Bayesian processing of non-identified models poses no serious difficulty as long as the posterior distribution is well defined. This is the case for a proper prior like

$$\pi(\beta, \sigma^2) \propto \sigma^{-4} \exp\{-1/\sigma^2\} \exp\{-\beta^2/50\}$$

that corresponds to a normal distribution on β and a gamma distribution on σ^{-2} . While the posterior distribution on (β, σ) is not a standard distribution, it is available up to a normalising constant. Therefore, it can be directly processed via

an MCMC algorithm. In this case, we chose a Gibbs sampler that simulates β and σ^2 alternatively, from

$$\pi(\beta|\mathbf{x}, \mathbf{y}, \sigma) \propto \prod_{y_i=1} \Phi(x_i\beta/\sigma) \prod_{y_i=0} \Phi(-x_i\beta/\sigma) \times \pi(\beta)$$

and

$$\pi(\sigma^2|\mathbf{x}, \mathbf{y}, \beta) \propto \prod_{y_i=1} \Phi(x_i\beta/\sigma) \prod_{y_i=0} \Phi(-x_i\beta/\sigma) \times \pi(\sigma^2)$$

respectively. Since both of these conditional distributions are also non-standard, we replace the direct simulation by a one-dimensional Metropolis–Hastings step, using normal $\mathcal{N}(\beta^{(t)}, 1)$ and log-normal $\mathcal{LN}(\log \sigma^{(t)}, 0.04)$ random walk proposals, respectively. For a simulated dataset of 1000 points, the contour plot of the log-posterior distribution is given in Fig. 11.7, along with the last 1000 points of a corresponding MCMC sample after 100,000 iterations. This graph shows a very satisfactory repartition of the simulated parameters over the likelihood surface, with higher concentrations near the largest posterior regions. For another simulation, Fig. 11.8 details the first 500 steps, when started at $(\beta, \sigma^2) = (0.1, 4.0)$. Although each step contains both a β and a σ proposal, some moves are either horizontal or vertical: this corresponds to cases when either the β or the σ proposals have been rejected. Note also the fairly rapid convergence to a modal zone of the posterior distribution in this case.

Obviously, this is only a toy example and more realistic probit models do not fare so well with down-the-shelf random walk Metropolis–Hastings algorithms, as shown for instance in Nobile (1998) (see also Robert and Casella, 2004, Sect. 10.3.2).⁸

The difficulty inherent to random walk Metropolis–Hastings algorithms is the scaling of the proposal distribution: it must be adapted to the shape of the target distribution so that, in a reasonable number of steps, the whole support of this distribution can be visited. If the scale of the proposal is too small, this will not happen as the algorithm stays “too local” and, if there are several modes on the posterior, the algorithm may get trapped within one modal region because it cannot reach other modal regions with jumps of too small magnitude. The larger the dimension p is, the harder it is to set up the right scale, though, because

- (a) the curse of dimension implies that there are more and more empty regions in the space, that is, regions with zero posterior probability;
- (b) the knowledge and intuition about the modal regions get weaker and weaker;
- (c) the proper scaling involves a symmetric (p, p) matrix Ξ in the proposal $g((\theta - \theta')^T \Xi (\theta - \theta'))$. Even when the matrix Ξ is diagonal, it gets harder to scale as the

⁸ Even in the simple case of the probit model, MCMC algorithms do not always converge very quickly, as shown in Robert and Casella (2004, Chap. 14).

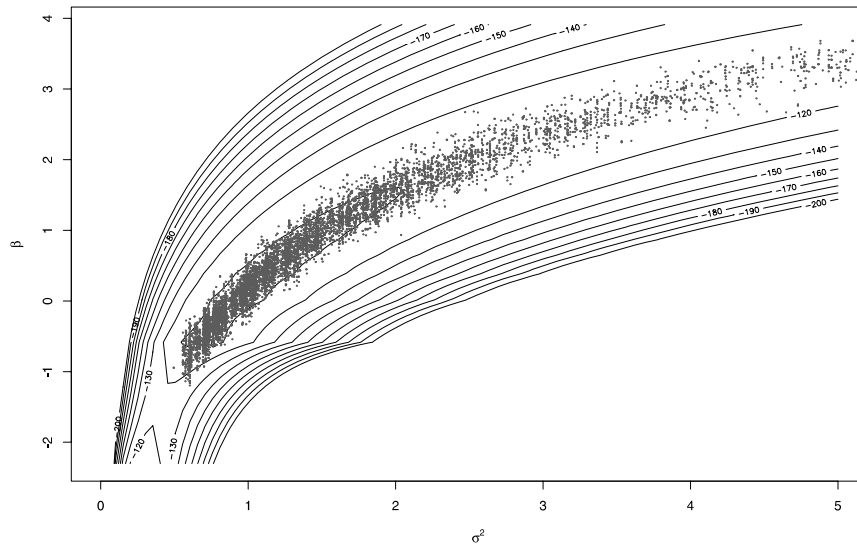


Figure 11.7. Contour plot of the log-posterior distribution for a probit sample of 1000 observations, along with 1000 points of an MCMC sample (Source: Robert and Casella, 2004)

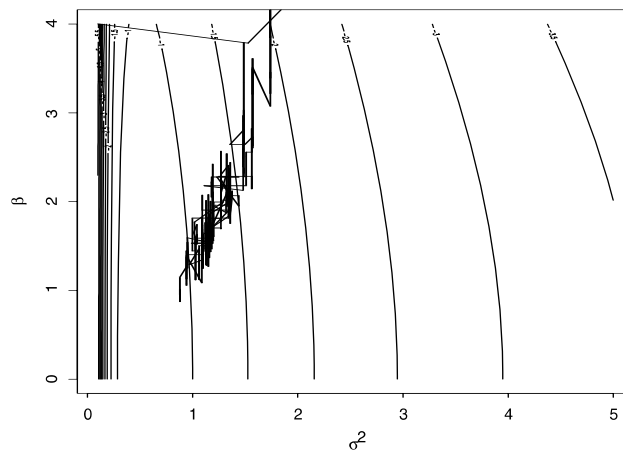


Figure 11.8. First 500 steps of the Metropolis–Hastings algorithm on the probit log-posterior surface, when started at $(\beta, \sigma^2) = (0.1, 4.0)$

dimension increases (unless one resorts to a Gibbs like implementation, where each direction is scaled separately).

Note also that the on-line scaling of the algorithm against the empirical acceptance rate is inherently flawed in that the attraction of a modal region may give a false sense of convergence and lead to a choice of too small a scale, simply because other modes will not be visited during the scaling experiment.

Gibbs Sampling and Latent Variable Models

11.4.2

The Gibbs sampler is a definitely attractive algorithm for Bayesian problems because it naturally fits the hierarchical structures so often found in such problems. “Natural” being a rather vague notion from a simulation point of view, it routinely happens that other algorithms fare better than the Gibbs sampler. Nonetheless, Gibbs sampler is often worth a try (possibly with other Metropolis–Hastings refinements at a later stage) in well-structured objects like Bayesian hierarchical models and more general graphical models.

A very relevant illustration is made of latent variable models, where the observational model is itself defined as a mixture model,

$$f(x|\theta) = \int_{\mathcal{Z}} f(x|z, \theta) g(z|\theta) dz .$$

Such models were instrumental in promoting the Gibbs sampler in the sense that they have the potential to make Gibbs sampling sound natural very easily. (See also Chap. II.3.) For instance, Tanner and Wong (1987) wrote a precursor article to Gelfand and Smith (1990) that designed specific two-stage Gibbs samplers for a variety of latent variable models. And many of the first applications of Gibbs sampling in the early 90’s were actually for models of that kind. The usual implementation of the Gibbs sampler in this case is to simulate the missing variables Z conditional on the parameters and reciprocally, as follows:

Latent Variable Gibbs Algorithm

At iteration t

1. Generate $z^{(t+1)} \sim g(z|\theta^{(t)})$
2. Generate $\theta^{(t+1)} \sim \pi(\theta|x, z^{(t+1)})$

While we could have used the probit case as an illustration (Example 11), as done in Chap. II.3, we choose to pick the case of mixtures (Example 6) as a better setting.

Example 15: (*Continuation of Example 6*) The natural missing data structure of a mixture of distribution is historical. In one of the first mixtures to be ever studied by Bertillon, in 1863, a bimodal structure on the height of conscripts in south eastern France (Doubs) can be explained by the mixing of two populations of military conscripts, one from the plains and one from the mountains (or hills). Therefore, in the analysis of data from distributions of the form

15

$$\sum_{i=1}^k p_i f(x|\theta_i) ,$$

a common missing data representation is to associate with each observation x_j a missing multinomial variable $z_j \sim \mathcal{M}_k(1; p_1, \dots, p_k)$ such that $x_j|z_j = i \sim f(x|\theta_i)$.

In heterogeneous populations made of several homogeneous subgroups or sub-populations, it makes sense to interpret z_j as the index of the population of origin of x_j , which has been lost in the observational process.

However, mixtures are also customarily used for density approximations, as a limited dimension proxy to non-parametric approaches. In such cases, the components of the mixture and even the number k of components in the mixture are often meaningless for the problem to be analysed. But this distinction between natural and artificial completion (by the z_j 's) is lost to the MCMC sampler, whose goal is simply to provide a Markov chain that converges to the posterior as stationary distribution. Completion is thus, from a simulation point of view, a mean to generate such a chain.

The most standard Gibbs sampler for mixture models (Diebolt and Robert, 1994) is thus based on the successive simulation of the z_j 's and of the θ_i 's, conditional on one another and on the data:

1. Generate $z_j|\theta, x_j$ ($j = 1, \dots, n$)
2. Generate $\theta_i|\mathbf{x}, \mathbf{z}$ ($i = 1, \dots, k$)

Given that the density f is most often from an exponential family, the simulation of the θ_i 's is generally straightforward.

As an illustration, consider the case of a normal mixture with two components, with equal known variance and fixed weights,

$$p \mathcal{N}(\mu_1, \sigma^2) + (1-p) \mathcal{N}(\mu_2, \sigma^2) . \quad (11.10)$$

Assume in addition a normal $\mathcal{N}(0, 10\sigma^2)$ prior on both means μ_1 and μ_2 . It is easy to see that μ_1 and μ_2 are independent, given (\mathbf{z}, \mathbf{x}) , and the respective conditional distributions are

$$\mathcal{N}\left(\sum_{z_i=j} x_i / (0.1 + n_j), \sigma^2 / (0.1 + n_j)\right) ,$$

where n_j denotes the number of z_i 's equal to j . Even more easily, it comes that the conditional posterior distribution of \mathbf{z} given (μ_1, μ_2) is a product of binomials, with

$$\begin{aligned} P(Z_i = 1 | x_i, \mu_1, \mu_2) \\ = \frac{p \exp\{-(x_i - \mu_1)^2 / 2\sigma^2\}}{p \exp\{-(x_i - \mu_1)^2 / 2\sigma^2\} + (1-p) \exp\{-(x_i - \mu_2)^2 / 2\sigma^2\}} . \end{aligned}$$

Figure 11.9 illustrates the behavior of the Gibbs sampler in that setting, with a simulated dataset of 100 points from the $0.7\mathcal{N}(0, 1) + 0.3\mathcal{N}(2.7, 1)$ distribution. The representation of the MCMC sample after 5000 iterations is quite in agreement with the posterior surface, represented via a grid on the (μ_1, μ_2) space and some contours. The sequence of consecutive steps represented on the left graph also shows that the mixing behavior is satisfactory, since the jumps are in scale with the modal region of the posterior.

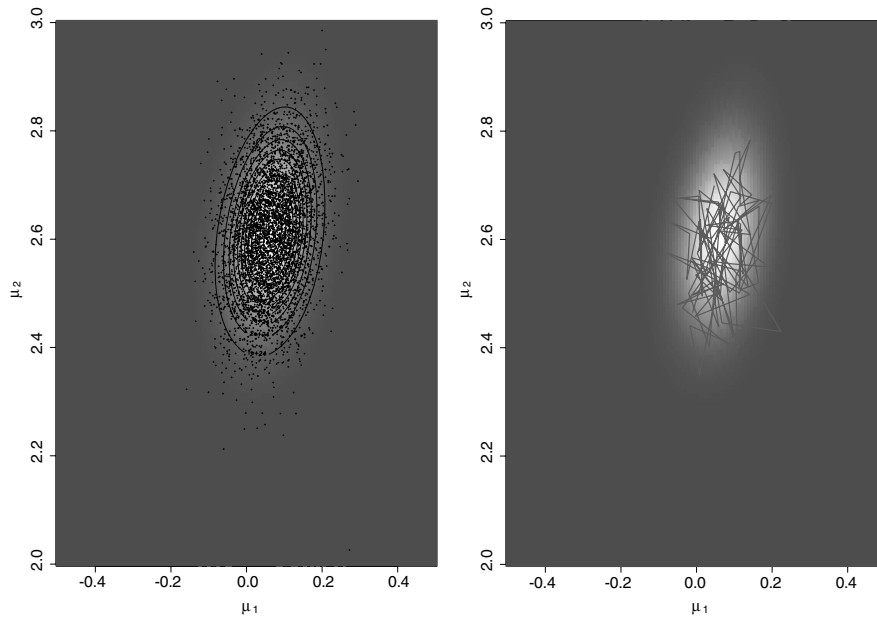


Figure 11.9. Gibbs sample of 5000 points for the mixture posterior (*left*) and path of the last 100 consecutive steps (*right*) against the posterior surface (Source: Robert and Casella, 2004)

This experiment gives a wrong sense of safety, though, because it does not point out the fairly large dependence of the Gibbs sampler to the initial conditions, already signaled in Diebolt and Robert (1994) under the name of *trapping states*. Indeed, the conditioning of (μ_1, μ_2) on z implies that the new simulations of the means will remain very close to the previous values, especially if there are many observations, and thus that the new allocations z will not differ much from the previous allocations. In other words, to see a significant modification of the allocations (and thus of the means) would require a very very large number of iterations. Figure 11.10 illustrates this phenomenon for the same sample as in Fig. 11.9, for a wider scale: there always exists a second mode in the posterior distribution, which is much lower than the first mode located around $(0, 2.7)$. Nonetheless, a Gibbs sampler initialized close to the second and lower mode will not be able to leave the vicinity of this (irrelevant) mode, even after a large number of iterations. The reason is as given above: to jump to the other mode, a majority of z_j 's would need to change simultaneously and the probability of such a jump is too close to 0 to let the event occur.⁹

⁹ It is quite interesting to see that the mixture Gibbs sampler suffers from the same pathology as the EM algorithm, although this is not surprising given that it is based on the same completion scheme.

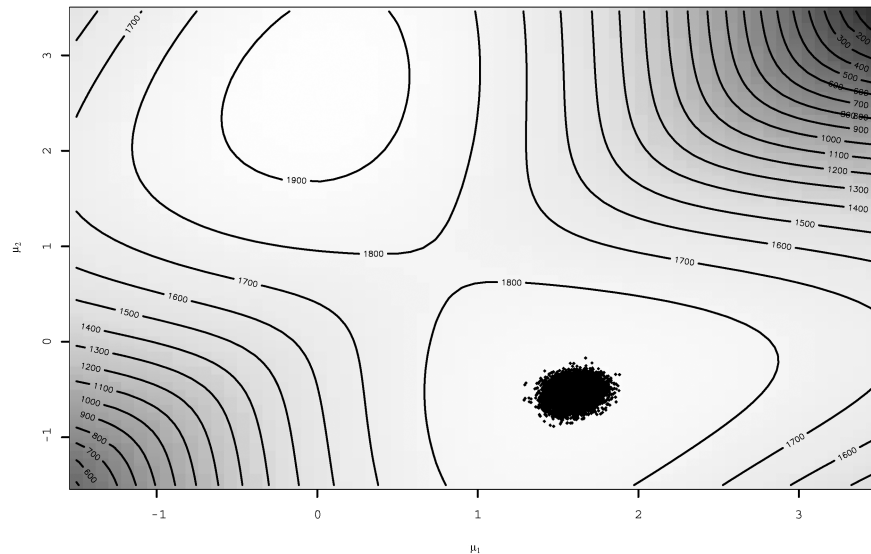


Figure 11.10. Posterior surface and corresponding Gibbs sample for the two mean mixture model, when initialized close to the second and lower mode, based on 10,000 iterations (Source: Robert and Casella, 2004)

This example illustrates quite convincingly that, while the completion is natural from a model point of view (since it is a part of the definition of the model), it does not necessarily transfer its utility for the simulation of the posterior. Actually, when the missing variable model allows for a closed form likelihood, as is the case for mixtures, probit models (Examples 11 and 14) and even hidden Markov models (see Cappé and Rydén, 2004), the whole range of the MCMC technology can be used as well. The appeal of alternatives like random walk Metropolis–Hastings schemes is that they remain in a smaller dimension space, since they avoid the completion step(s), and that they are not restricted in the range of their moves.¹⁰

16

Example 16: (Continuation of Example 15) Given that the likelihood of a sample (x_1, \dots, x_n) from the mixture distribution (11.10) can be computed in $O(2n)$ time, a regular random walk Metropolis–Hastings algorithm can be used in this setup. Figure 11.11 shows how quickly this algorithm escapes the attraction of the poor mode, as opposed to the Gibbs sampler of Fig. 11.10: within a few iterations of the algorithm, the chain drifts over the poor mode and converges almost deterministically to the proper region of the posterior surface. The random walk is based on $\mathcal{N}(\mu_i^{(t)}, 0.04)$ proposals, although other scales would work as well but would

¹⁰ This wealth of possible alternatives to the completion Gibbs sampler is a mixed blessing in that their range, for instance the scale of the random walk proposals, needs to be scaled properly to avoid inefficiencies.

require more iterations to reach the proper model regions. For instance, a scale of 0.005 in the Normal proposal above needs close to 5000 iterations to attain the main mode.

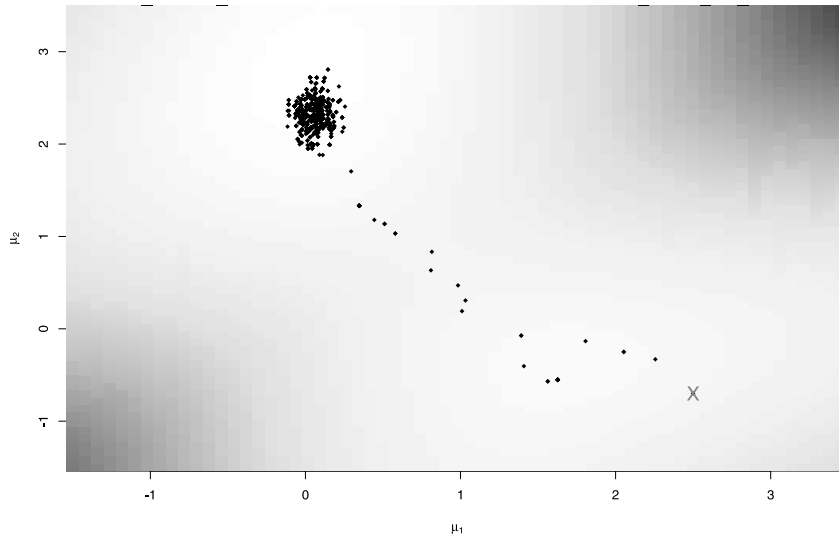


Figure 11.11. Track of a 1000 iteration random walk Metropolis–Hastings sample on the posterior surface, the starting point is indicated by a cross. (The scale of the random walk is 0.2.)

The secret of a successful MCMC implementation in such latent variable models is to maintain the distinction between latency in models and latency in simulation (the later being often called use of *auxiliary variables*). When latent variables can be used with adequate mixing of the resulting chain and when the likelihood cannot be computed in a closed form (as in hidden semi-Markov models, Cappé et al., 2004), a Gibbs sampler is a still simple solution that is often easy to simulate from. Adding well-mixing random walk Metropolis–Hastings steps in the simulation scheme cannot hurt the overall mixing of the chain (Robert and Casella, 2004, Chap. 13), especially when several scales can be used at once (see Sect. 11.5). A final word is that the completion can be led in an infinity of ways and that several of these should be tried or used in parallel to increase the chances of success.

Reversible Jump Algorithms for Variable Dimension Models

11.4.3

As described in Sect. 11.2.3, model choice is computationally different from testing in that it considers at once a (much) wider range of models \mathfrak{M}_i and parameter spaces Θ_i . Although early approaches could only go through a pedestrian pairwise

comparison, a more adequate perspective is to envision the model index i as part of the parameter to be estimated, as in (11.3). The (computational) difficulty is that we are then dealing with a possibly infinite space that is the collection of unrelated sets: how can we then simulate from the corresponding distribution?¹¹

The MCMC solution proposed by Green (1995) is called *reversible jump MCMC*, because it is based on a *reversibility* constraint on the transitions between the sets Θ_i . In fact, the only real difficulty compared with previous developments is to validate moves (or *jumps*) between the Θ_i 's, since proposals restricted to a given Θ_i follow from the usual (fixed-dimensional) theory. Furthermore, *reversibility* can be processed at a local level: since the model indicator μ is a integer-valued random variable, we can impose reversibility for each pair (k_1, k_2) of possible values of μ . The idea at the core of reversible jump MCMC is then to supplement each of the spaces Θ_{k_1} and Θ_{k_2} with adequate artificial spaces in order to create a *bijection* between them. For instance, if $\dim(\Theta_{k_1}) > \dim(\Theta_{k_2})$ and if the move from Θ_{k_1} to Θ_{k_2} can be represented by a *deterministic* transformation of $\theta^{(k_1)}$

$$\theta^{(k_2)} = T_{k_1 \rightarrow k_2}(\theta^{(k_1)}) ,$$

Green (1995) imposes a *dimension matching* condition which is that the opposite move from Θ_{k_2} to Θ_{k_1} is concentrated on the curve

$$\{\theta^{(k_1)} : \theta^{(k_2)} = T_{k_1 \rightarrow k_2}(\theta^{(k_1)})\} .$$

In the general case, if $\theta^{(k_1)}$ is completed by a simulation $u_1 \sim g_1(u_1)$ into $(\theta^{(k_1)}, u_1)$ and $\theta^{(k_2)}$ by $u_2 \sim g_2(u_2)$ into $(\theta^{(k_2)}, u_2)$ so that the mapping between $(\theta^{(k_1)}, u_1)$ and $(\theta^{(k_2)}, u_2)$ is a bijection,

$$(\theta^{(k_2)}, u_2) = T_{k_1 \rightarrow k_2}(\theta^{(k_1)}, u_1) , \quad (11.11)$$

the probability of acceptance for the move from model \mathfrak{M}_{k_1} to model \mathfrak{M}_{k_2} is then

$$\min \left(\frac{\pi(k_2, \theta^{(k_2)})}{\pi(k_1, \theta^{(k_1)})} \frac{\pi_{21} g_2(u_2)}{\pi_{12} g_1(u_1)} \left| \frac{\partial T_{k_1 \rightarrow k_2}(\theta^{(k_1)}, u_1)}{\partial (\theta^{(k_1)}, u_1)} \right|, 1 \right) ,$$

involving

- the Jacobian of the transform $T_{k_1 \rightarrow k_2}$,
- the probability π_{ij} of choosing a jump to \mathcal{M}_{k_j} while in \mathcal{M}_{k_i} , and
- g_i , the density of u_i .

¹¹ Early proposals to solve the varying dimension problem involved saturation schemes where all the parameters for all models were updated deterministically (Carlin and Chib, 1995), but they do not apply for an infinite collection of models and they need to be precisely calibrated to achieve a sufficient amount of moves between models.

The acceptance probability for the reverse move is based on the inverse ratio if the move from \mathfrak{M}_{k_2} to \mathfrak{M}_{k_1} also satisfies (11.11) with $u_2 \sim g_2(u_2)$.¹²

The pseudo-code representation of Green’s algorithm is thus as follows:

Green’s Algorithm

At iteration t , if $x^{(t)} = (m, \theta^{(m)})$,

1. Select model \mathfrak{M}_n with probability π_{mn}
2. Generate $u_{mn} \sim \varphi_{mn}(u)$
3. Set $(\theta^{(n)}, v_{nm}) = T_{m \rightarrow n}(\theta^{(m)}, u_{mn})$
4. Take $x^{(t+1)} = (n, \theta^{(n)})$ with probability

$$\min \left(\frac{\pi(n, \theta^{(n)})}{\pi(m, \theta^{(m)})} \frac{\pi_{nm} \varphi_{nm}(v_{nm})}{\pi_{mn} \varphi_{mn}(u_{mn})} \left| \frac{\partial T_{m \rightarrow n}(\theta^{(m)}, u_{mn})}{\partial (\theta^{(m)}, u_{mn})} \right|, 1 \right),$$

and take $x^{(t+1)} = x^{(t)}$ otherwise.

As for previous methods, the implementation of this algorithm requires a certain skillfulness in picking the right proposals and the appropriate scales. This art of reversible jump MCMC is illustrated on the two following examples, extracted from Robert and Casella (2004, Sect. 14.2.3).

Example 17: (Continuation of Example 6) If we consider for model \mathfrak{M}_k the k component normal mixture distribution,

17

$$\sum_{j=1}^k p_{jk} \mathcal{N}(\mu_{jk}, \sigma_{jk}^2),$$

moves between models involve changing the number of components in the mixture and thus adding new components or removing older components or yet again changing several components. As in Richardson and Green (1997), we can restrict the moves when in model \mathfrak{M}_k to only models \mathfrak{M}_{k+1} and \mathfrak{M}_{k-1} . The simplest solution is to use a birth-and-death process: The *birth step* consists in adding a new normal component in the mixture generated from the prior and the *death step* is the opposite, removing one of the k components at random. In this case, the corresponding birth acceptance probability is

$$\begin{aligned} & \min \left(\frac{\pi_{(k+1)k}}{\pi_{k(k+1)}} \frac{(k+1)!}{k!} \frac{\pi_{k+1}(\theta_{k+1})}{\pi_k(\theta_k) (k+1) \varphi_{k(k+1)}(u_{k(k+1)})}, 1 \right) \\ & = \min \left(\frac{\pi_{(k+1)k}}{\pi_{k(k+1)}} \frac{\rho(k+1)}{\rho(k)} \frac{\ell_{k+1}(\theta_{k+1}) (1-p_{k+1})^{k-1}}{\ell_k(\theta_k)}, 1 \right), \end{aligned}$$

¹² For a simple proof that the acceptance probability guarantees that the stationary distribution is $\pi(k, \theta^{(k)})$, see Robert and Casella (2004, Sect. 11.2.2).

where ℓ_k denotes the likelihood of the k component mixture model \mathfrak{M}_k and $\rho(k)$ is the prior probability of model \mathfrak{M}_k .¹³

While this proposal can work well in some setting, as in Richardson and Green (1997) when the prior is calibrated against the data, it can also be inefficient, that is, leading to a high rejection rate, if the prior is vague, since the birth proposals are not tuned properly. A second proposal, central to the solution of Richardson and Green (1997), is to devise more local jumps between models, called *split* and *combine* moves, since a new component is created by splitting an existing component into two, under some moment preservation conditions, and the reverse move consists in combining two existing components into one, with symmetric constraints that ensure reversibility. (See, e.g., Robert and Casella, 2004, for details.)

Figures 11.12–11.14 illustrate the implementation of this algorithm for the so-called Galaxy dataset used by Richardson and Green (1997) (see also Roeder, 1992), which contains 82 observations on the speed of galaxies. On Fig. 11.12, the MCMC output on the number of components k is represented as a histogram on k , and the corresponding sequence of k 's. The prior used on k is a uniform distribution on $\{1, \dots, 20\}$: as shown by the lower plot, most values of k are explored by the reversible jump algorithm, but the upper bound does not appear to be restrictive since the $k^{(t)}$'s hardly ever reach this upper limit. Figure 11.13 illustrates the fact that conditioning the output on the most likely value of k (3 here) is possible. The nine graphs in this figure show the joint variation of the three types of parameters, as well as the stability of the Markov chain over the 1,000,000 iterations: the cumulated averages are quite stable, almost from the start.

The density plotted on top of the histogram in Fig. 11.14 is another good illustration of the inferential possibilities offered by reversible jump algorithms, as a case of *model averaging*: this density is obtained as the average over iterations t of

$$\sum_{j=1}^{k^{(t)}} p_{jk}^{(t)} \mathcal{N} \left(\mu_{jk}^{(t)}, \left(\sigma_{jk}^{(t)} \right)^2 \right),$$

which approximates the posterior expectation $\mathbb{E}[f(y|\theta)|\mathbf{x}]$, where \mathbf{x} denotes the data x_1, \dots, x_{82} .

18

Example 18: (*Continuation of Example 3*) For the $AR(p)$ model of Example 3, the best way to include the stationarity constraints is to use the lag-polynomial representation

$$\prod_{i=1}^p (1 - \lambda_i B) X_t = \varepsilon_t, \quad \varepsilon_t \sim \mathcal{N}(0, \sigma^2),$$

¹³ In the birth acceptance probability, the factorials $k!$ and $(k+1)!$ appear as the numbers of ways of ordering the k and $k+1$ components of the mixtures. The ratio cancels with $1/(k+1)$, which is the probability of selecting a particular component for the death step.

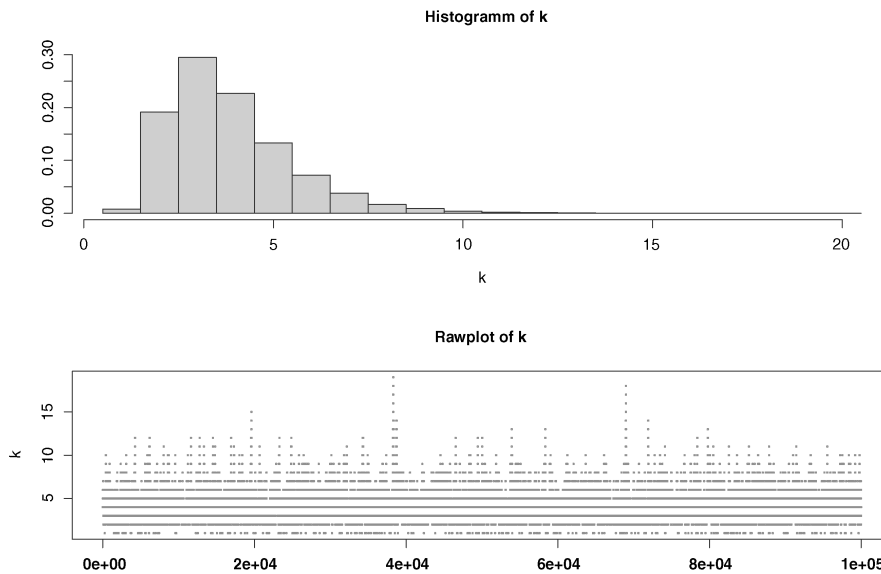


Figure 11.12. Histogram and raw plot of 100,000 k 's produced by a reversible jump MCMC algorithm for the Galaxy dataset

of model \mathfrak{M}_p , and to constrain the inverse roots, λ_i , to stay within the unit circle if complex and within $[-1, 1]$ if real (see, e.g. Robert, 2001, Sect. 4.5.2). The associated uniform priors for the real and complex roots λ_j is

$$\pi_p(\boldsymbol{\lambda}) = \frac{1}{\lfloor p/2 \rfloor + 1} \prod_{\lambda_i \in \mathbb{R}} \frac{1}{2} \mathbb{I}_{|\lambda_i| < 1} \prod_{\lambda_i \notin \mathbb{R}} \frac{1}{\pi} \mathbb{I}_{|\lambda_i| < 1},$$

where $\lfloor p/2 \rfloor + 1$ is the number of different values of r_p . This factor must be included within the posterior distribution when using reversible jump since it does not vanish in the acceptance probability of a move between models \mathfrak{M}_p and \mathfrak{M}_q . Otherwise, this results in a modification of the prior probability of each model.

Once again, a simple choice is to use a birth-and-death scheme where the birth moves either create a real or two conjugate complex roots. As in the birth-and-death proposal for Example 17, the acceptance probability simplifies quite dramatically since it is for instance

$$\min \left(\frac{\pi_{(p+1)p}}{\pi_{p(p+1)}} \frac{(r_p + 1)!}{r_p!} \frac{\lfloor p/2 \rfloor + 1}{\lfloor (p + 1)/2 \rfloor + 1} \frac{\ell_{p+1}(\boldsymbol{\theta}_{p+1})}{\ell_p(\boldsymbol{\theta}_p)}, 1 \right)$$

in the case of a move from \mathfrak{M}_p to \mathfrak{M}_{p+1} . (As for the above mixture example, the factorials are related to the possible choices of the created and the deleted roots.)

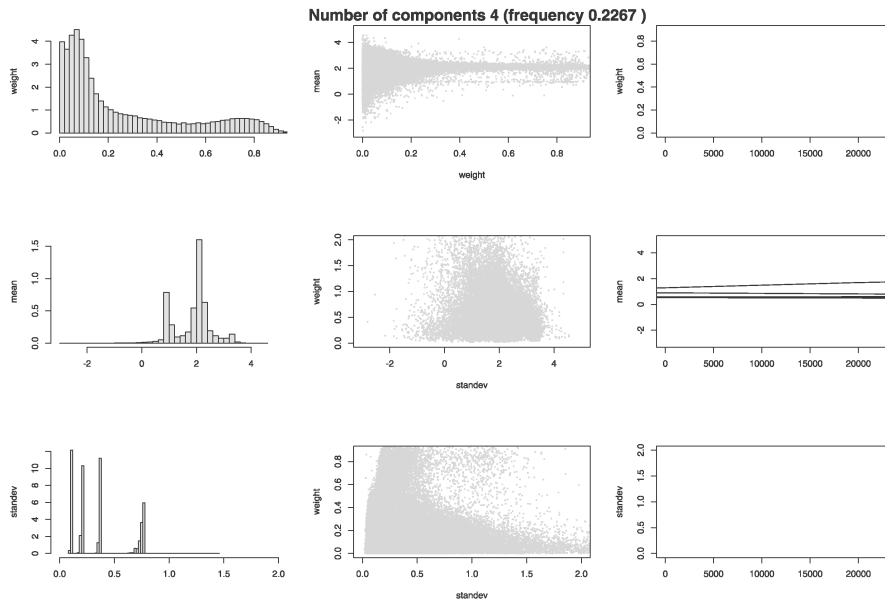


Figure 11.13. Reversible jump MCMC output on the parameters of the model \mathcal{M}_3 for the Galaxy dataset, obtained by conditioning on $k = 3$. The *left column* gives the histogram of the weights, means, and variances; the *middle column* the scatterplot of the pairs weights-means, means-variances, and variances-weights; the *right column* plots the cumulated averages (over iterations) for the weights, means, and variances

Figure 11.15 presents some views of the corresponding reversible jump MCMC algorithm. Besides the ability of the algorithm to explore a range of values of k , it also shows that Bayesian inference using these tools is much richer, since it can, for instance, condition on or average over the order k , mix the parameters of different models and run various tests on these parameters. A last remark on this graph is that both the order and the value of the parameters are well estimated, with a characteristic trimodality on the histograms of the θ_i 's, even when conditioning on k different from 3, the value used for the simulation.

11.5 More Monte Carlo Methods

While MCMC algorithms considerably expanded the range of applications of Bayesian analysis, they are not, by any means, the end of the story! Further developments are taking place, either at the fringe of the MCMC realm or far away from it. We indicate below a few of the directions in Bayesian computational Statistics, omitting many more that also are of interest...

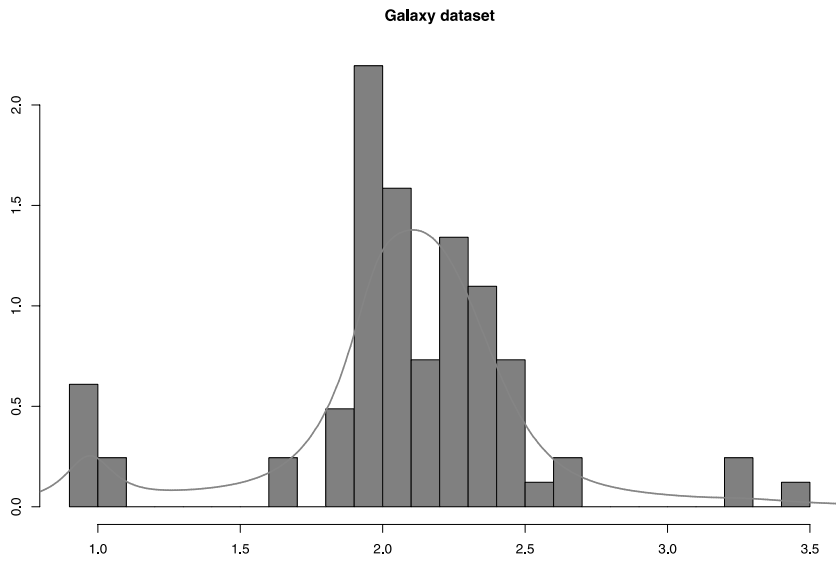


Figure 11.14. Fit of the dataset by the averaged density, $\mathbb{E}[f(y|\theta)|x]$

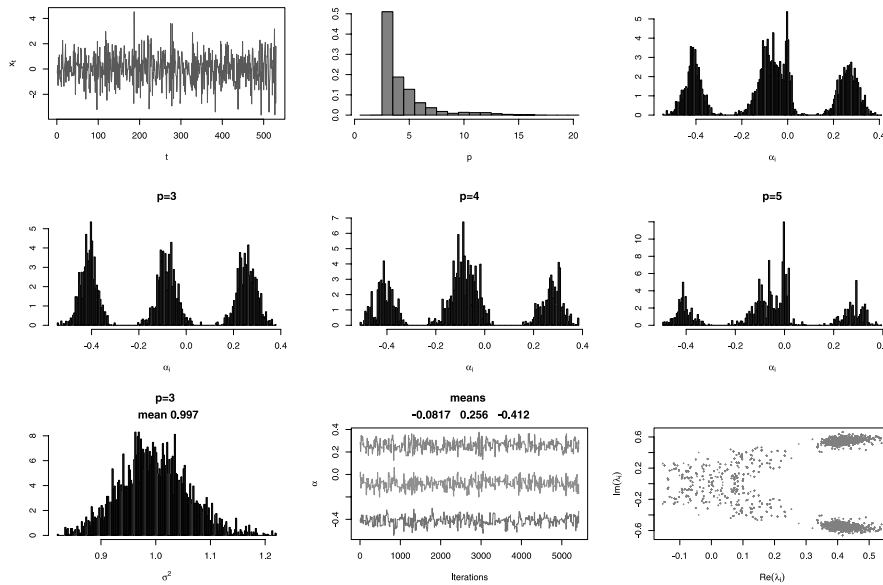


Figure 11.15. Output of a reversible jump algorithm based on an AR(3) simulated dataset of 530 points (upper left) with true parameters θ_i ($-0.1, 0.3, -0.4$) and $\sigma = 1$. The first histogram is associated with k , the following histograms are associated with the θ_i 's, for different values of k , and of σ^2 . The final graph is a scatterplot of the complex roots (for iterations where there were complex roots). The one before last graph plots the evolution over the iterations of $\theta_1, \theta_2, \theta_3$ (Source: Robert 2003)

11.5.1 Adaptivity for MCMC Algorithms

Given the range of situations where MCMC applies, it is unrealistic to hope for a *generic* MCMC sampler that would function in every possible setting. The more generic proposals like random-walk Metropolis–Hastings algorithms are known to fail in large dimension and disconnected supports, because they take too long to explore the space of interest (Neal, 2003). The reason for this impossibility theorem is that, in realistic problems, the complexity of the distribution to simulation is the very reason why MCMC is used! So it is difficult to ask for a prior opinion about this distribution, its support or the parameters of the proposal distribution used in the MCMC algorithm: intuition is close to void in most of these problems.

However, the performances of off-the-shelf algorithms like the random-walk Metropolis–Hastings scheme bring information about the distribution of interest and, as such, should be incorporated in the design of better and more powerful algorithms. The problem is that we usually miss the time to train the algorithm on these previous performances and are looking for the Holy Grail of automated MCMC procedures! While it is natural to think that the information brought by the first steps of an MCMC algorithm should be used in later steps, there is a severe catch: using the whole past of the “chain” implies that this is not a Markov chain any longer. Therefore, usual convergence theorems do not apply and the validity of the corresponding algorithms is questionable. Further, it may be that, in practice, such algorithms do degenerate to point masses because of a too rapid decrease in the variation of their proposal.

19 Example 19: (*Continuation of Example 9*) For the t -distribution sample, we could fit a normal proposal from the empirical mean and variance of the previous values of the chain,

$$\mu_t = \frac{1}{t} \sum_{i=1}^t \theta^{(i)} \quad \text{and} \quad \sigma_t^2 = \frac{1}{t} \sum_{i=1}^t (\theta^{(i)} - \mu_t)^2 .$$

This leads to a Metropolis–Hastings algorithm with acceptance probability

$$\prod_{j=2}^n \left[\frac{v + (x_j - \theta^{(t)})^2}{v + (x_j - \xi)^2} \right]^{-(v+1)/2} \frac{\exp - (\mu_t - \theta^{(t)})^2 / 2\sigma_t^2}{\exp - (\mu_t - \xi)^2 / 2\sigma_t^2} ,$$

where ξ is the proposed value from $\mathcal{N}(\mu_t, \sigma_t^2)$. The invalidity of this scheme (because of the dependence on the whole sequence of $\theta^{(i)}$'s till iteration t) is illustrated in Fig. 11.16: when the range of the initial values is too small, the sequence of $\theta^{(i)}$'s cannot converge to the target distribution and concentrates on too small a support. But the problem is deeper, because even when the range of the simulated values is correct, the (long-term) dependence on past values modifies the distribution of the sequence. Figure 11.17 shows that, for an initial variance of 2.5, there is a bias in the histogram, even after 25,000 iterations and stabilisation of the empirical mean and variance.

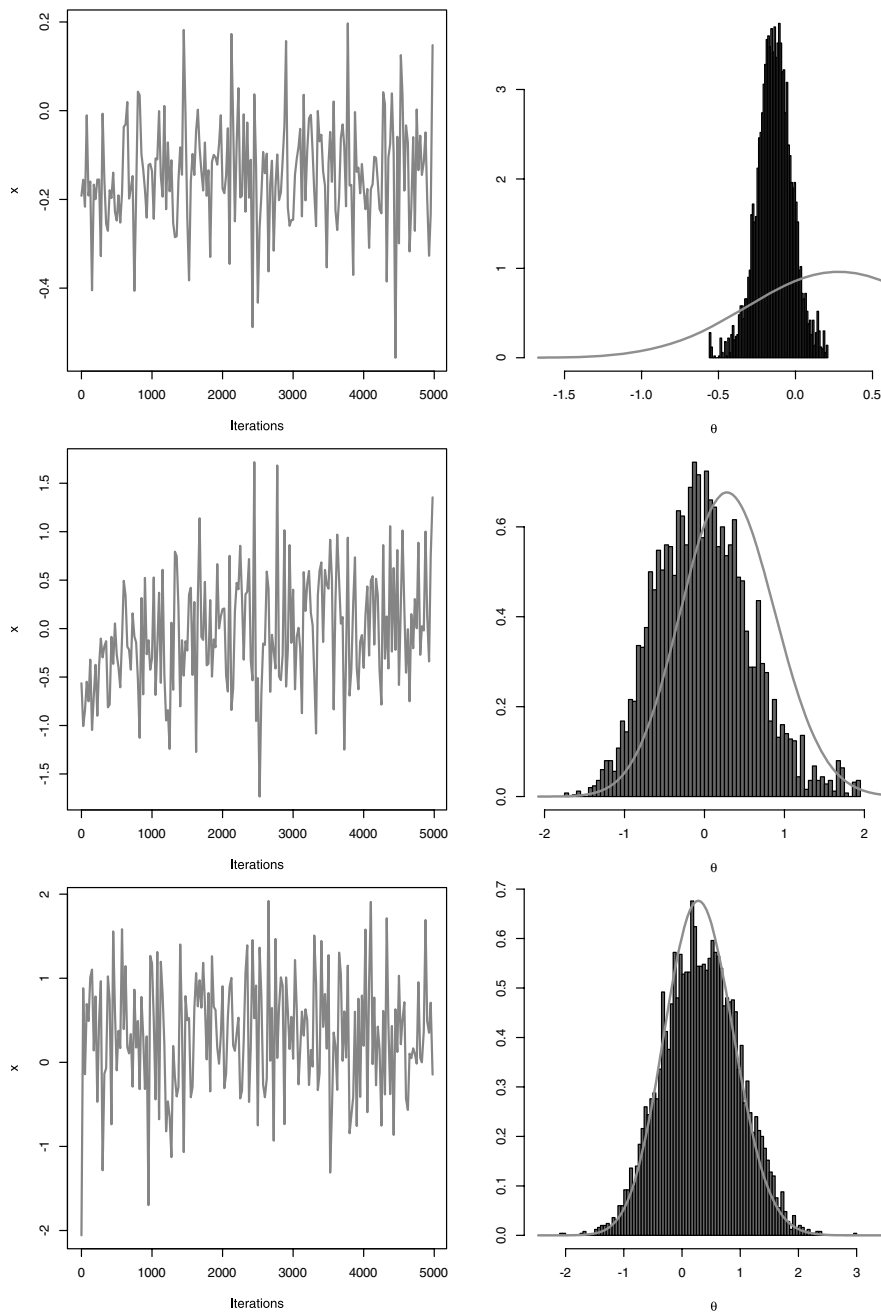


Figure 11.16. Output of the adaptive scheme for the t -distribution posterior with a sample of $10x_j \sim \mathcal{T}_3$ and initial variances of (top) 0.1, (middle) 0.5, and (bottom) 2.5. The left column plots the sequence of $\theta^{(i)}$'s while the right column compares its histogram against the true posterior distribution (with a different scale for the upper graph)

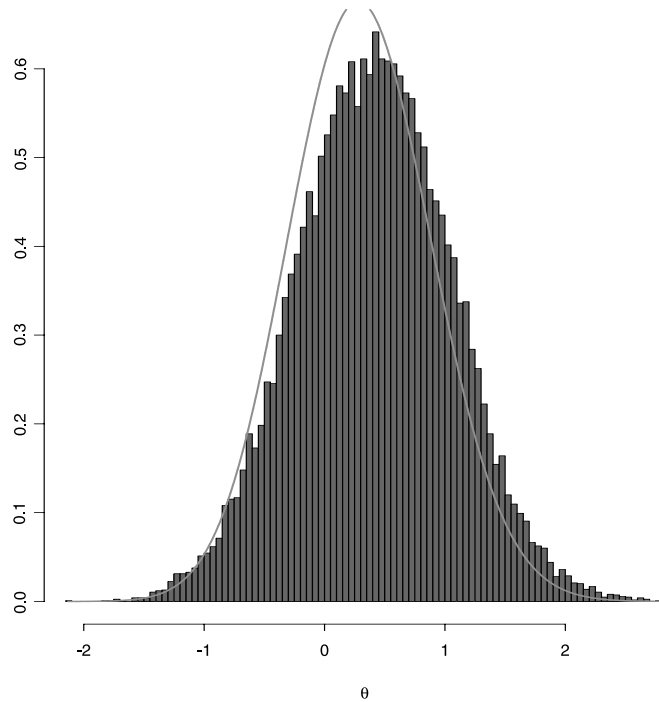


Figure 11.17. Comparison of the distribution of an adaptive scheme sample of 25,000 points with initial variance of 2.5 and of the target distribution

Even though the Markov chain is converging *in distribution* to the target distribution (when using a proper, i.e. time-homogeneous updating scheme), using past simulations to create a non-parametric approximation to the target distribution does not work either. Figure 11.18 shows for instance the output of an adaptive scheme in the setting of Example 19 when the proposal distribution is the Gaussian kernel based on earlier simulations. A very large number of iterations is not sufficient to reach an acceptable approximation of the target distribution.

The overall message is thus that one should not *constantly* adapt the proposal distribution on the past performances of the simulated chain. Either the adaptation must cease after a period of *burnin* (not to be taken into account for the computations of expectations and quantities related to the target distribution), or the adaptive scheme must be theoretically assessed on its own right. This latter path is not easy and only a few examples can be found (so far) in the literature. See, e.g., Gilks et al. (1998) who use regeneration to create block independence and preserve Markovianity on the paths rather than on the values, Haario et al. (1999,2001) who derive a proper adaptation scheme in the spirit of Example 19 by using a ridge-like correction to the empirical variance, and Andrieu and Robert (2001) who propose a more general framework of valid adaptivity based on stochastic optimisation and the Robbin-Monro algorithm. (The latter actually embeds the

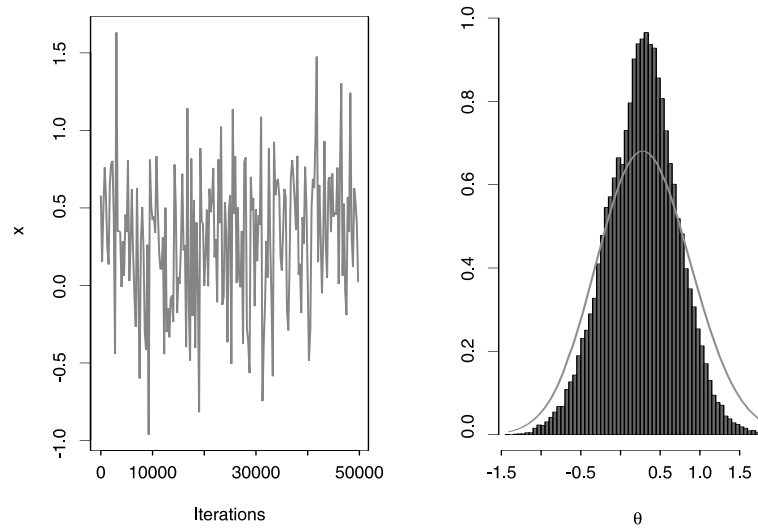


Figure 11.18. Sample produced by 50,000 iterations of a nonparametric adaptive MCMC scheme and comparison of its distribution with the target distribution

chain of interest $\theta^{(t)}$ in a larger chain $(\theta^{(t)}, \xi^{(t)}, \partial^{(t)})$ that also includes the parameter of the proposal distribution as well as the gradient of a performance criterion.)

Population Monte Carlo

11.5.2

To reach acceptable adaptive algorithms, while avoiding an extended study of their theoretical properties, a better alternative is to leave the structure of Markov chains and to consider *sequential* or *population* Monte Carlo methods (Iba (2000), Cappé et al. (2004)) that have much more in common with importance sampling than with MCMC. They are inspired from *particle systems* that were introduced to handle rapidly changing target distributions like those found in signal processing and imaging (Gordon et al., 1993; Shephard and Pitt, 1997; Doucet et al., 2001) but primarily handle fixed but complex target distributions by building a sequence of increasingly better proposal distributions.¹⁴ Each iteration of the population Monte Carlo (PMC) algorithm thus produces a sample approximately simulated from the target distribution but the iterative structure allows for adaptivity toward the target distribution. Since the validation is based on importance sampling principles, dependence on the past samples can be arbitrary *and* the approximation to the target is valid (unbiased) at *each iteration* and does not require convergence times nor stopping rules.

¹⁴ The “sequential” denomination in the sequential Monte Carlo methods thus refers to the algorithmic part, not to the statistical part.

If t indexes the iteration and i the sample point, consider proposal distributions q_{it} that simulate the $x_i^{(t)}$'s and associate to each $x_i^{(t)}$ an importance weight

$$\rho_i^{(t)} = \pi(x_i^{(t)}) / q_{it}(x_i^{(t)}) , \quad i = 1, \dots, n .$$

Approximations of the form

$$\mathfrak{J}_t = \frac{1}{n} \sum_{i=1}^n \rho_i^{(t)} h(x_i^{(t)})$$

are then unbiased estimators of $\mathbb{E}^\pi[h(X)]$, even when the importance distribution q_{it} depends on the entire past of the experiment. Indeed, if ζ denotes the vector of past random variates that contribute to q_{it} , and $g(\zeta)$ its *arbitrary* distribution, we have

$$\iint \frac{\pi(x)}{q_{it}(x|\zeta)} h(x) q_{it}(x) dx g(\zeta) d\zeta = \iint h(x) \pi(x) dx g(\zeta) d\zeta = \mathbb{E}^\pi[h(X)] .$$

Furthermore, assuming that the variances

$$\text{var}(\rho_i^{(t)} h(x_i^{(t)}))$$

exist for every $1 \leq i \leq n$, we have

$$\text{var}(\mathfrak{J}_t) = \frac{1}{n^2} \sum_{i=1}^n \text{var}(\rho_i^{(t)} h(x_i^{(t)})) ,$$

due to the canceling effect of the weights $\rho_i^{(t)}$.

Since, usually, the density π is unnormalized, we use instead

$$\rho_i^{(t)} \propto \frac{\pi(x_i^{(t)})}{q_{it}(x_i^{(t)})} , \quad i = 1, \dots, n ,$$

scaled so that the $\rho_i^{(t)}$'s sum up to 1. In this case, the unbiasedness is lost, although it approximately holds. In fact, the estimation of the normalizing constant of π improves with each iteration t , since the overall average

$$\bar{\omega}_t = \frac{1}{tn} \sum_{\tau=1}^t \sum_{i=1}^n \frac{\pi(x_i^{(\tau)})}{q_{i\tau}(x_i^{(\tau)})}$$

is convergent. Therefore, as t increases, $\bar{\omega}_t$ contributes less and less to the variability of \mathfrak{J}_t .

Since the above establishes that a simulation scheme based on sample dependent proposals is fundamentally a specific kind of importance sampling, the

following algorithm is validated by the same principles as regular importance sampling:

Population Monte Carlo Algorithm

For $t = 1, \dots, T$

1. For $i = 1, \dots, n$,
 - (1) Select the generating distribution $q_{it}(\cdot)$
 - (2) Generate $x_i^{(t)} \sim q_{it}(x)$
 - (3) Compute $\rho_i^{(t)} = \pi(x_i^{(t)})/q_{it}(x_i^{(t)})$
2. Normalize the $\rho_i^{(t)}$'s to sum up to 1
3. Resample n values from the $x_i^{(t)}$'s with replacement, using the weights $\rho_i^{(t)}$, to create the sample $(x_1^{(t)}, \dots, x_n^{(t)})$

Step (1) is singled out because it is the central property of the PMC algorithm, namely that adaptivity can be extended to the individual level and that the q_{it} 's can be picked based on the performances of the previous $q_{i(t-1)}$'s or even on all the previously simulated samples, if storage allows. For instance, the q_{it} 's can include large tails proposals as in the *defensive sampling* strategy of Hesterberg (1998), to ensure finite variance. Similarly, Warnes' (2001) non-parametric Gaussian kernel approximation can be used as a proposal.¹⁵ (See also Stavropoulos and Titterton, 2001 *smooth bootstrap* as an earlier example of PMC algorithm.)

The major difference between the PMC algorithm and earlier proposals in the particle system literature is that past dependent moves as those of Gilks and Berzuini (2001) remain within the MCMC framework, with Markov transition kernels with stationary distribution equal to π .

Example 20: (*Continuation of Example 15*) We consider here the implementation of the PMC algorithm in the case of the normal mixture (11.10). As in Example 16, a PMC sampler can be efficiently implemented *without* the (Gibbs) augmentation step, using normal random walk proposals based on the previous sample of $\boldsymbol{\mu} = (\mu_1, \mu_2)$'s. Moreover, the difficulty inherent to random walks, namely the selection of a "proper" scale, can be bypassed because of the adaptivity of the PMC algorithm. Indeed, the proposals can be associated with a range of variances ν_k ($1 \leq k \leq K$) ranging from, e.g., 10^3 down to 10^{-3} . At each step of the algorithm, the new variances can be selected proportionally to the performances of the scales ν_k on the previous iterations. For instance, a scale can be chosen proportionally to its *non-degeneracy rate* in the previous iteration, that is, the percentage of points

20

¹⁵ Using a Gaussian non-parametric kernel estimator amounts to (a) sampling from the $x_i^{(t)}$'s with equal weights and (b) using a normal random walk move from the selected $x_i^{(t)}$, with standard deviation equal to the bandwidth of the kernel.

generated with the scale ν_k that survived after resampling.¹⁶ The weights are then of the form

$$\rho_j \propto \frac{f(\mathbf{x} | (\mu_1)_j^{(i)}, (\mu_2)_j^{(i)}) \pi((\mu_1)_j^{(i)}, (\mu_2)_j^{(i)})}{\varphi((\mu_1)_j^{(i)} | (\mu_1)_j^{(i-1)}, \nu_k) \varphi((\mu_2)_j^{(i)} | (\mu_2)_j^{(i-1)}, \nu_k)},$$

where $\varphi(q|s, \nu)$ is the density of the normal distribution with mean s and variance ν at the point q .

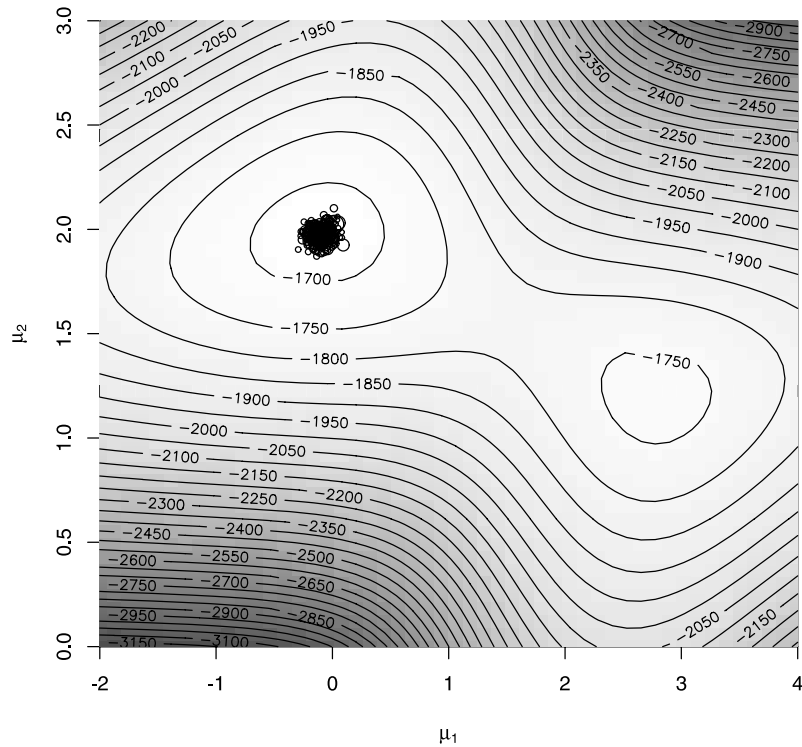


Figure 11.19. Representation of the log-posterior distribution with the PMC weighted sample after 30 iterations (the weights are proportional to the circles at each point) (Source: Cappé et al., 2004)

Compared with an MCMC algorithm in the same setting (see Examples 15 and 16), the main feature of this algorithm is its ability to deal with multiscale proposals in an unsupervised manner. The upper row of Fig. 11.21 produces the frequencies of the five variances ν_k used in the proposals along iterations: The two largest variances ν_k most often have a zero survival rate, but sometimes experience bursts of survival. In fact, too large a variance mostly produces points

¹⁶ When the survival rate of a proposal distribution is null, in order to avoid the complete removal of a given scale ν_k , the corresponding number r_k of proposals with that scale is set to a positive value, like 1% of the sample size.

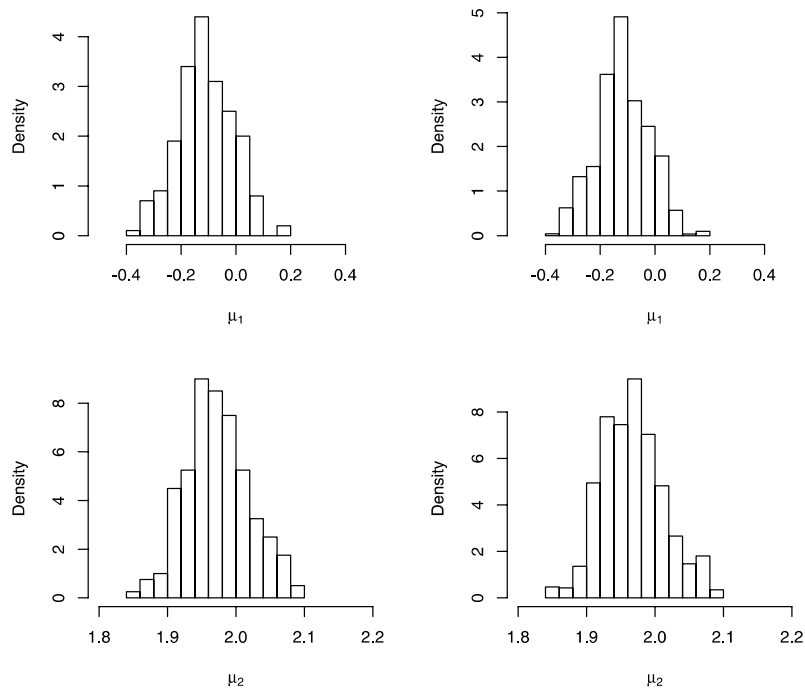


Figure 11.20. Histograms of the PMC sample: sample at iteration 5 (*left*) before resampling and (*right*) after resampling

that are irrelevant for the posterior distribution, but once in a while a point $\theta_j^{(t)}$ gets close to one of the modes of the posterior. When this occurs, the corresponding ρ_j is large and $\theta_j^{(t)}$ is thus heavily resampled. The upper right graph shows that the other proposals are rather evenly sampled along iterations. The influence of the variation in the proposals on the estimation of the means μ_1 and μ_2 can be seen on the middle and lower panels of Fig. 11.21. First, the cumulative averages quickly stabilize over iterations, by virtue of the general importance sampling proposal. Second, the corresponding variances take longer to stabilize but this is to be expected, given the regular reappearance of subsamples with large variances.

In comparison with Figs. 11.10 and 11.11, Fig. 11.19 shows that the sample produced by the PMC algorithm is quite in agreement with the modal zone of the posterior distribution. The second mode, which is much lower, is not preserved in the sample after the first iteration. Figure 11.20 also shows that the weights are quite similar, with no overwhelming weight in the sample.

The generality in the choice of the proposal distributions q_{it} is obviously due to the abandonment of the MCMC framework. The difference with an MCMC

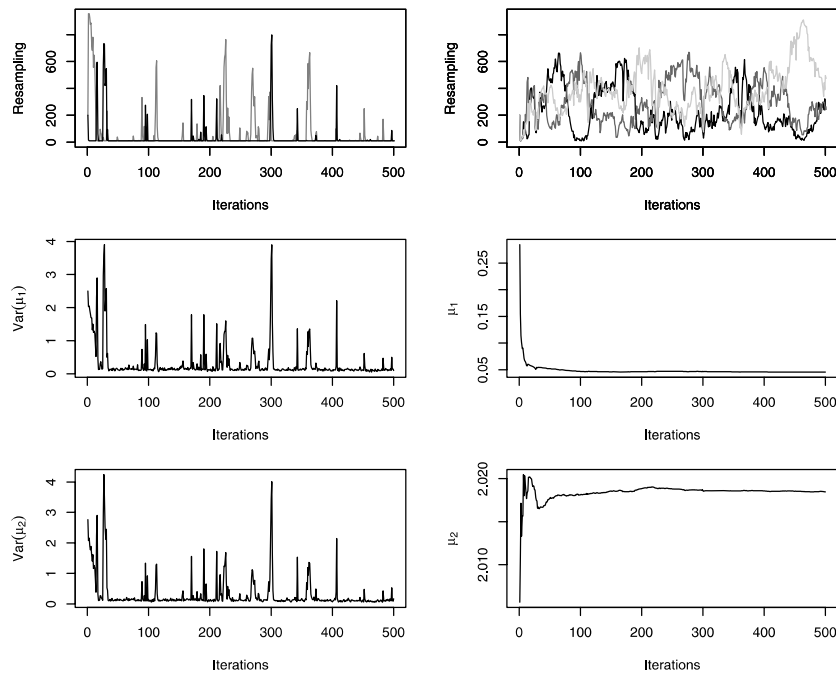


Figure 11.21. Performances of the mixture PMC algorithm for 1000 observations from a $0.2\mathcal{N}(0, 1) + 0.8\mathcal{N}(2, 1)$ distribution, with $\theta = 1$, $\lambda = 0.1$, $v_k = 5, 2, 0.1, 0.05, 0.01$, and a population of 1050 particles: (*upper left*) Number of resampled points for the variances $v_1 = 5$ (*darker*) and $v_2 = 2$; (*upper right*) Number of resampled points for the other variances, $v_3 = 0.1$ is the *darkest one*; (*middle left*) Variance of the simulated μ_1 's along iterations; (*middle right*) Cumulated average of the simulated μ_1 's over iterations; (*lower left*) Variance of the simulated μ_2 's along iterations; (*lower right*) Cumulated average of the simulated μ_2 's over iterations (Source: Cappé et al., 2004)

framework is not simply a theoretical advantage: as seen in Sect. 11.5.1, proposals based on the whole past of the chain do not often work. Even algorithms validated by MCMC steps may have difficulties: in one example of Cappé et al. (2004), a Metropolis–Hastings scheme does not work well, while a PMC algorithm based on the same proposal produces correct answers.

11.6 Conclusion

This short overview of the problems and solutions considered for Bayesian Statistics is nothing but an introduction to the game: there are much more complex problems than those illustrated above and much more advanced techniques than those presented in these pages. The reader is then encouraged to enter the literature on the topic, maybe with other introductory surveys like Cappé and Robert

(2000) and Andrieu et al. (2004), but mostly through books like Chen et al. (2000), Doucet et al. (2001), Liu (2001), Green et al. (2003) and Robert and Casella (2004).

We have not mentioned so far entries to Bayesian softwares like winBUGS, developed by the MRC Unit in Cambridge (Gilks et al., 1994; Spiegelhalter et al., 1999), Ox (Doornik et al., 2002), BATS (Pole et al., 1994), BACC (Geweke, 1999) and the Minitab package of Albert (1996), which all cover some aspects of Bayesian computing. Obviously, these packages require some expertise from the user and are thus more difficult of use than the classical open source or commercial softwares like R, Splus, Statgraphics, StatXact, SPSS or SAS. In other words, they are not *black boxes* that could be used by laymen with no statistical background. But this entrance fee to the use of Bayesian softwares is inevitable, given the versatile nature of Bayesian analysis: since it offers much more variability than standard inferential procedures, through the choice of prior distributions and loss functions for instance, it also requires more input from the user! And, once these preliminary steps have been overcome, the programming involved in a software like winBUGS is rather limited and certainly not harder than writing a code in R or Matlab.

As stressed in this chapter, computational issues are central to the design and implementation of Bayesian analysis. The new era opened by the MCMC methodology has brought much more freedom in the use of Bayesian methods, as reflected by the increase of Bayesian studies in applied Statistics. As usually the case, a strong increase in the use of a methodology also sees a corresponding increase in its misuse! Inconsistent data-dependent priors and improper posteriors are sometimes appearing in studies and, more generally, the assessment of prior modelling (or even of MCMC convergence) are rarely conducted with sufficient care. This is somehow a price to pay for the wider range of Bayesian studies, while the improvement of corresponding software should bring more guidelines and warnings about these misuses of Bayesian analysis.

References

- Abowd, J., Kramarz, F., and Margolis, D. (1999). High-wage workers and high-wage firms. *Econometrica*, 67:251–333.
- Albert, J. (1996). *Bayesian Computation Using Minitab*. Wadsworth Publishing Company.
- Andrieu, C., Doucet, A., and Robert, C. (2004). Computational advances for and from Bayesian analysis. *Statistical Science*. (to appear).
- Andrieu, C. and Robert, C.P. (2001). Controlled Markov chain Monte Carlo methods for optimal sampling. Technical Report 0125, Université Paris Dauphine.
- Bauwens, L. and Richard, J.F. (1985). A 1-1 Poly- t random variable generator with application to Monte Carlo integration. *J. Econometrics*, 29:19–46.
- Cappé, O., Guillin, A., Marin, J.M., and Robert, C.P. (2004). Population Monte Carlo. *J. Comput. Graph. Statist.* (to appear).

- Cappé, O. and Robert, C.P. (2000). MCMC: Ten years and still running! *J. American Statist. Assoc.*, 95(4):1282–1286.
- Cappé, O. and Rydén, T. (2004). *Hidden Markov Models*. Springer-Verlag.
- Carlin, B.P. and Chib, S. (1995). Bayesian model choice through Markov chain Monte Carlo. *J. Roy. Statist. Soc. (Ser. B)*, 57(3):473–484.
- Chen, M.H., Shao, Q.M., and Ibrahim, J.G. (2000). *Monte Carlo Methods in Bayesian Computation*. Springer-Verlag.
- Diebolt, J. and Robert, C.P. (1994). Estimation of finite mixture distributions by Bayesian sampling. *J. Royal Statist. Soc. Series B*, 56:363–375.
- Doornik, J.A., Hendry, D.F., and Shephard, N. (2002). Computationally-intensive econometrics using a distributed matrix-programming language. *Philo. Trans. Royal Society London*, 360:1245–1266.
- Doucet, A., de Freitas, N., and Gordon, N. (2001). *Sequential Monte Carlo Methods in Practice*. Springer-Verlag, New York.
- Gelfand, A.E. and Smith, A.F.M. (1990). Sampling based approaches to calculating marginal densities. *J. American Statist. Assoc.*, 85:398–409.
- Geweke, J. (1999). Using simulation methods for Bayesian econometric models: Inference, development, and communication (with discussion and rejoinder). *Econometric Reviews*, 18:1–126.
- Gilks, W.R. and Berzuini, C. (2001). Following a moving target—Monte Carlo inference for dynamic Bayesian models. *J. Royal Statist. Soc. Series B*, 63(1):127–146.
- Gilks, W.R., Roberts, G.O., and Sahu, S.K. (1998). Adaptive Markov chain Monte Carlo. *J. American Statist. Assoc.*, 93:1045–1054.
- Gilks, W.R., Thomas, A., and Spiegelhalter, D.J. (1994). A language and program for complex Bayesian modelling. *The Statistician*, 43:169–178.
- Gordon, N., Salmond, J., and Smith, A.F.M. (1993). A novel approach to non-linear/non-Gaussian Bayesian state estimation. *IEEE Proceedings on Radar and Signal Processing*, 140:107–113.
- Green, P.J. (1995). Reversible jump MCMC computation and Bayesian model determination. *Biometrika*, 82(4):711–732.
- Green, P.J., Hjort, N.L., and Richardson, S. (2003). *Highly Structured Stochastic Systems*. Oxford University Press, Oxford, UK.
- Haario, H., Saksman, E., and Tamminen, J. (1999). Adaptive proposal distribution for random walk Metropolis algorithm. *Computational Statistics*, 14(3):375–395.
- Haario, H., Saksman, E., and Tamminen, J. (2001). An adaptive Metropolis algorithm. *Bernoulli*, 7(2):223–242.
- Hesterberg, T. (1998). Weighted average importance sampling and defensive mixture distributions. *Technometrics*, 37:185–194.
- Iba, Y. (2000). Population-based Monte Carlo algorithms. *Trans. Japanese Soc. Artificial Intell.*, 16(2):279–286.
- Jeffreys, H. (1961). *Theory of Probability (3rd edition)*. Oxford University Press, Oxford, 1939 edition.
- Liu, J.S. (2001). *Monte Carlo Strategies in Scientific Computing*. Springer-Verlag, New York, NY.

- McCullagh, P. and Nelder, J. (1989). *Generalized Linear Models*. Chapman and Hall.
- Meng, X.L. and Wong, W.H. (1996). Simulating ratios of normalizing constants via a simple identity: a theoretical exploration. *Statist. Sinica*, 6:831–860.
- Metropolis, N. and Ulam, S. (1949). The Monte Carlo method. *J. American Statist. Assoc.*, 44:335–341.
- Neal, R.M. (2003). Slice sampling (with discussion). *Ann. Statist.*, 31:705–767.
- Nobile, A. (1998). A hybrid Markov chain for the Bayesian analysis of the multinomial probit model. *Statistics and Computing*, 8:229–242.
- Pole, A., West, M., and Harrison, P.J. (1994). *Applied Bayesian Forecasting and Time Series Analysis*. Chapman-Hall, New York.
- Richardson, S. and Green, P.J. (1997). On Bayesian analysis of mixtures with an unknown number of components (with discussion). *J. Royal Statist. Soc. Series B*, 59:731–792.
- Robert, C.P. (2001). *The Bayesian Choice*. Springer-Verlag, second edition.
- Robert, C.P. and Casella, G. (1999). *Monte Carlo Statistical Methods*. Springer-Verlag, New York, NY.
- Robert, C.P. and Casella, G. (2004). *Monte Carlo Statistical Methods*. Springer-Verlag, New York, second edition. (to appear).
- Roeder, K. (1992). Density estimation with confidence sets exemplified by superclusters and voids in galaxies. *J. American Statist. Assoc.*, 85:617–624.
- Shephard, N. and Pitt, M.K. (1997). Likelihood analysis of non-Gaussian measurement time series. *Biometrika*, 84:653–668.
- Spiegelhalter, D.J., Best, N.G., Carlin, B.P., and van der Linde, A. (2002). Bayesian measures of model complexity and fit. *J. Royal Statistical Society Series B*, 64(3):583–639.
- Spiegelhalter, D.J., Thomas, A., and Best, N.G. (1999). *WinBUGS Version 1.2 User Manual*. Cambridge.
- Stavropoulos, P. and Titterton, D.M. (2001). Improved particle filters and smoothing. In Doucet, A., deFreitas, N., and Gordon, N., editors, *Sequential MCMC in Practice*. Springer-Verlag.
- Tanner, M. and Wong, W. (1987). The calculation of posterior distributions by data augmentation. *J. American Statist. Assoc.*, 82:528–550.
- Von Neumann, J. (1951). Various techniques used in connection with random digits. *J. Resources of the National Bureau of Standards – Applied Mathematics Series*, 12:36–38.

Computational Methods in Survival Analysis

III.12

Toshinari Kamakura

12.1	<i>Introduction</i>	768
	Nonparametric Model.....	768
	Parametric Models.....	769
12.2	<i>Estimation of Shape or Power Parameter</i>	772
12.3	<i>Regression Models</i>	774
	The Score Test	774
	Evaluation of Estimators in the Cox Model	776
	Approximation of Partial Likelihood	776
12.4	<i>Multiple Failures and Counting Processes</i>	779
	Intensity Function.....	779
	Multiple Counting Processes	780
	Power Law Model	782
	Models Suitable for Conditional Estimation	782

Survival analysis is widely used in the fields of medical science, pharmaceuticals, reliability and financial engineering, and many others to analyze positive random phenomena defined by event occurrences of particular interest. In the reliability field, we are concerned with the time to failure of some physical component such as an electronic device or a machine part. This article briefly describes statistical survival techniques developed recently from the standpoint of statistical computational methods focussing on obtaining the good estimates of distribution parameters by simple calculations based on the first moment and conditional likelihood for eliminating nuisance parameters and approximation of the likelihoods. The method of partial likelihood (Cox, 1972, 1975) was originally proposed from the view point of conditional likelihood for avoiding estimating the nuisance parameters of the baseline hazards for obtaining simple and good estimates of the structure parameters. However, in case of heavy ties of failure times calculating the partial likelihood does not succeed. Then the approximations of the partial likelihood have been studied, which will be described in the later section and a good approximation method will be explained. We believe that the better approximation method and the better statistical model should play an important role in lessening the computational burdens greatly.

12.1 Introduction

Let T be a positive random variable with density function $f(t)$ and distribution function $F(t)$. The survival function $S(t)$ is then defined as

$$S(t) = 1 - F(t) = \Pr\{T > t\},$$

and the hazard function or hazard rate as

$$\lambda(t) = \lim_{h \rightarrow 0} \frac{\Pr\{t < T \leq t + h | T > t\}}{h}.$$

The hazard function can also be expressed as

$$\lambda(t) = \frac{f(t)}{S(t)}. \quad (12.1)$$

The right-hand side (RHS) of (12.1) becomes

$$\frac{f(t)}{S(t)} = -\frac{d}{dt} \log S(t),$$

and inversely

$$S(t) = \exp \left\{ -\int_0^t \lambda(u) du \right\}. \quad (12.2)$$

Nonparametric Model

12.1.1

We assume that the observed data set consists of failure or death times t_i and censoring indicators δ_i , $i = 1, \dots, n$. The indicator δ is unity for the case of failure and zero for censoring. The censoring scheme is an important concept in survival analysis in that one can observe partial information associated with the survival random variable. This is due to some limitations such as loss to follow-up, drop-out, termination of the study, and others.

The Kaplan–Meier method (Kaplan and Meier, 1958) is currently the standard for estimating the nonparametric survival function. For the case of a sample without any censoring observations, the estimate exactly corresponds to the derivation from the empirical distribution. The dataset can be arranged in table form, i.e.,

Table 12.1. Failure time data

Failure times	t_1	t_2	\dots	t_i	\dots	t_k
Number of failures	d_1	d_2	\dots	d_i	\dots	d_k
Number of individuals of risk set	n_1	n_2	\dots	n_i	\dots	n_k

where, t_i is the i -th order statistic when they are arranged in ascending order for distinct failure times, d_i is the number of failures at the time of t_i , and n_i is the number of survivors at time $t_i - 0$. Under this notation the Kaplan–Meier estimate becomes

$$\widehat{S}(t) = \prod_{j:t_j < t} \left(1 - \frac{d_j}{n_j} \right). \quad (12.3)$$

The standard error of the Kaplan–Meier estimate is

$$SE \{ \widehat{S}(t) \} = [\widehat{S}(t)] \left\{ \sum_{j:t_j < t} \frac{d_j}{n_j(n_j - d_j)} \right\}^{1/2}. \quad (12.4)$$

The above formula is called “Greenwood’s formula” described by Greenwood (1926).

Parametric Models

12.1.2

The most important and widely-used models in survival analysis are exponential, Weibull, log-normal, log-logistic, and gamma distributions. The first two models will be introduced for later consideration. The exponential distribution is simplistic and easy to handle, being similar to a standard distribution in some respects, while the Weibull distribution is a generalization of the exponential distribution and allows inclusion of many types of shapes. Their density functions are

$$f(t; \lambda) = \lambda e^{-\lambda t} \quad (\lambda, t > 0) \quad (12.5)$$

$$f(t; m, \eta) = \frac{m}{\eta} \left(\frac{t}{\eta}\right)^{m-1} \exp\left\{-\left(\frac{t}{\eta}\right)^m\right\} \quad (m, \eta, t > 0), \quad (12.6)$$

where the parameter λ is sometimes called the failure rate in reliability engineering. Two models may include additional threshold parameters, or guarantee times. Let γ be this threshold parameter. The Weibull density function then becomes

$$f(t; m, \eta, \gamma) = \frac{m}{\eta} \left(\frac{t-\gamma}{\eta}\right)^{m-1} \exp\left\{-\left(\frac{t-\gamma}{\eta}\right)^m\right\} \quad (m, \eta, \gamma, t > 0). \quad (12.7)$$

Here, note that in the case of $m = 1$, the Weibull probability density function is exactly the exponential density function placing $\lambda = 1/\eta$, and that we cannot observe any failure times before threshold time ($t < \gamma$) or an individual cannot die before this time.

As the Weibull distribution completely includes the exponential distribution, only the Weibull model will be discussed further. The Weibull distribution is widely used in reliability and biomedical engineering because of goodness of fit to data and ease of handling. The main objective in lifetime analysis sometimes involves (1) estimation of a few parameters which define the Weibull distribution, and (2) evaluation of the effects of some environmental factors on lifetime distribution using regression techniques. Inference on the quantiles of the distribution has been previously studied in detail (Johnson et al., 1994).

The maximum likelihood estimate (MLE) is well known, yet it is not expressed explicitly in closed form. Accordingly, some iterative computational methods are used. Menon (Menon (1963)) provided a simple estimator of $1/m$, being a consistent estimate of $1/m$, with a bias that tends to vanish as the sample size increases. Later, Cohen (Cohen, 1965; Cohen and Whitten, 1988) presented a practically useful chart for obtaining a good first approximation to the shape parameter m using the property that the coefficient of variation of the Weibull distribution is a function of the shape parameter m , i.e., it does not depend on η . This is described as follows.

Let T be a random variable with probability density function (12.6), the r th moment around the origin is then calculated as

$$E[T^r] = \eta^r \Gamma\left(1 + \frac{r}{m}\right).$$

Here $\Gamma(\cdot)$ is the complete gamma function. From this, the first two moments obtained are the mean life and variance, i.e.,

$$E[T] = \eta \Gamma\left(1 + \frac{1}{m}\right),$$

$$\text{Var}[T] = \eta^2 \left\{ \Gamma\left(1 + \frac{2}{m}\right) - \Gamma^2\left(1 + \frac{1}{m}\right) \right\}.$$

Considering that the coefficient of variation

$$CV = \sqrt{(\text{Var}[T])/E[T]}$$

does not depend on the parameter η allows obtaining simple and robust moment estimates, which may be the initial values of the maximum likelihood calculations. Dubey (1967) studied the behavior of the Weibull distribution in detail based on these moments, concluding that the Weibull distribution with shape parameter $m = 3.6$ is relatively similar to the normal distribution.

Regarding the three-parameter Weibull described by (12.7), Cohen and Whitten (1988) suggested using the method of moments equations, noting that

$$\begin{aligned} E[T] &= \gamma + \eta\Gamma_1(m), \\ \text{Var}[T] &= \eta^2 \{ \Gamma_2(m) - \Gamma_1^2(m) \}, \\ E[X_{(1)}] &= \gamma + \frac{\eta}{n^{1/m}}\Gamma_1(m), \end{aligned}$$

and equating them to corresponding samples, where $\Gamma_r(m) = \Gamma(1 + r/m)$.

As for obtaining an inference on the parameter of the mean parameter $\mu = E(T)$, this has not yet been investigated and will now be discussed. When one would like to estimate μ , use of either the MLE or the standard sample mean is best for considering the case of an unknown shape parameter. This is true because the asymptotic relative efficiency of the sample mean to the MLE is calculated as

$$\begin{aligned} \text{ARE}(\bar{T}) &= \frac{n\text{Avar}(\bar{\mu})}{n\text{Avar}(\bar{T})} \\ &= \frac{6}{m^2\pi^2} \cdot \frac{1}{\text{CV}^2} \left[\frac{\pi^2}{6} + \{c - 1 + \psi(1 + 1/m)\}^2 \right], \end{aligned} \tag{12.8}$$

where c is Euler’s constant, $\psi(\cdot)$ a digamma function, $\bar{\mu}$ the MLE, and \bar{T} the sample mean.

Table 12.2 gives the ARE with respect to various values of m . Note the remarkably high efficiency of the sample mean, especially for $m \geq 0.5$, where more than 90% efficiency is indicated. The behavior of $\text{ARE}(\bar{T})$ for $m > 1$ is that $\text{ARE}(\bar{T})$ has a local minimum 0.9979 at $m = 1.7884$ and a local maximum 0.9986 at $m = 3.1298$, and that for the larger m , $\text{ARE}(\bar{T})$ monotonically decreases in m and the infimum of $\text{ARE}(\bar{T})$ is given in $m \rightarrow \infty$;

$$\lim_{m \rightarrow \infty} \text{ARE}(\bar{T}) = \frac{6(\pi^2 + 6)}{\pi^4} \cong 0.9775. \tag{12.9}$$

When m is known and tends to infinity, the behavior of $\text{ARE}(\bar{T})$ is as follows:

$$\lim_{m \rightarrow \infty} \frac{1}{(m\text{CV})^2} = \frac{6}{\pi^2} \cong 0.6079. \tag{12.10}$$

Table 12.2. ARE of the sample mean to the MLE

m	eff	m	eff	m	eff
0.1	0.0018	1.1	0.9997	2.1	0.9980
0.2	0.1993	1.2	0.9993	2.2	0.9981
0.3	0.5771	1.3	0.9988	2.3	0.9982
0.4	0.8119	1.4	0.9984	2.4	0.9983
0.5	0.9216	1.5	0.9981	2.5	0.9984
0.6	0.9691	1.6	0.9980	2.6	0.9984
0.7	0.9890	1.7	0.9979	2.7	0.9985
0.8	0.9968	1.8	0.9979	2.8	0.9985
0.9	0.9995	1.9	0.9979	2.9	0.9985
1.0	1.0000	2.0	0.9980	3.0	0.9986

A higher relative efficiency of the sample mean for unknown m is shown compared to known m . From a practical standpoint, the sample mean is easily calculated for a point estimation of the Weibull mean if no censored data are included. These results support the benefits of using the sample mean for the complete sample.

12.2

Estimation of Shape or Power Parameter

Let us now consider the class of the lifetime distributions, whose distribution functions are expressed by

$$F(t; \alpha, \gamma, \sigma) = G\left(\left(\frac{t - \gamma}{\sigma}\right)^\alpha\right), \quad (12.11)$$

where $G(\cdot)$ is also a distribution function. For the Weibull model, $G(t) = 1 - \exp(-t)$ is an exponential distribution. Nagatsuka and Kamakura (Nagatsuka and Kamakura, 2003, 2004) proposed a new method using the location-scale-free transformation of data set to estimate the power parameter in the Castillo–Hadi model (Castillo and Hadi, 1995). That is, let T_1, \dots, T_n be independently distributed according to the distribution function (12.11). Consider the W -transformation to be defined as

$$W_i = \frac{T_i - T_{(1)}}{T_{(n)} - T_{(1)}}, \quad (i = 2, \dots, n - 1), \quad (12.12)$$

where $T_{(k)}$ is the k -th order statistic of T_i 's. The new random variables W_i 's derived by this W -transformation are then free from location and scale parameter. The arithmetic mean of W_i 's gives the approximation to the original distribution of T . Let $V_i, i = 1, \dots, n$ be i.i.d. distributed with common distribution function $F_V(v)$,

and let the i -th order statistic $V_{(i)}$ have the marginal distribution function $F_{V_{(i)}}(v)$. Then

$$F_v(v) = \frac{1}{n} \sum_{i=1}^n F_{V_{(i)}}(v). \tag{12.13}$$

This equation indicates that the arithmetic mean of the marginal distributions of n order statistics is exactly the original distribution. In the case of the Castillo–Hadi Model, Nagatsuka and Kamakura (2004) provided a theorem regarding this approximation, i.e.,

Theorem 1 (Nagatsuka and Kamakura, 2004)

The mixture of the marginal distributions of $W_{(i)}$, $i = 2, \dots, n - 1$:

$$F^{(n)}(w) = \frac{1}{n-2} \sum_{i=2}^{n-1} F_{W_{(i)}}(w) \tag{12.14}$$

is the approximate distribution of W_i 's and the limiting distribution (12.14) is the power function distribution with parameter $1/\alpha$. That is

$$\lim_{n \rightarrow \infty} \frac{1}{n-2} \sum_{i=2}^{n-1} F_{W_{(i)}}(w) = w^{\frac{1}{\alpha}}, \quad 0 < w < 1.$$

In the case of the Weibull distribution, the marginal distribution of $W_{(i)}$ is calculated as

$$\begin{aligned} F_{W_{(i)}}(w) &= \Pr(W_{(i)} \leq w) \\ &= \Pr\left(\frac{T_{(i)} - T_{(1)}}{T_{(n)} - T_{(1)}} \leq w\right) \\ &= \int_0^\infty \int_u^\infty n(n-1)f(u)f(v) \left[\sum_{k=i-1}^{n-2} \binom{n-2}{k} \right. \\ &\quad \times \{F((1-w)u + wv) - F(u)\}^k \\ &\quad \left. \times \{F(v) - F((1-w)u + wv)\}^{n-k-2} \right] dv du \\ &= \int_0^1 \int_u^1 n(n-1) \sum_{k=i-1}^{n-2} \binom{n-2}{k} \times [1 - \exp\{-\alpha(w, m, u, v)\} - u]^k \\ &\quad \times [v - (1 - \exp\{-\alpha(w, m, u, v)\})]^{n-k-2}, \end{aligned} \tag{12.15}$$

where

$$\alpha(w, m, u, v) = \left[(1-w) \{-\log(1-u)\}^{\frac{1}{m}} + w \{-\log(1-v)\}^{\frac{1}{m}} \right]^m.$$

Calculations show that $F^{(n)}(w)$ has a first moment of

$$\begin{aligned} \mu_n(m) &= \int_0^{\infty} \{1 - F^{(n)}(w)\} dw \\ &= -\frac{1}{n-2} + \frac{n(n-1)}{m} \int_0^1 \int_u^1 (v-u)^{n-3} \\ &\quad \times \frac{\Gamma\left(\frac{1}{m}, -\log(1-u), -\log(1-v)\right)}{\{-\log(1-v)\}^{\frac{1}{m}} - \{-\log(1-u)\}^{\frac{1}{m}}} dv du. \end{aligned} \quad (12.16)$$

where $\Gamma(\cdot, \cdot, \cdot)$ is the incomplete generalized gamma function defined by

$$\Gamma(a, z_0, z_1) = \int_{z_0}^{z_1} t^{a-1} e^{-t} dt.$$

Now, an estimating of the shape parameter m is obtained by equating the theoretical population mean with sample mean of W -transformed W 's. Nagatsuka and Kamakura (2003) provided a table for obtaining estimates and concluded based on simulation studies that the robust estimate of m is possible without using any existing threshold parameter.

12.3

Regression Models

Survival analysis is now a standard statistical method for lifetime data. Fundamental and classical parametric distributions are also very important, but regression methods are very powerful to analyze the effects of some covariates on life lengths. Cox (1972) introduced a model for the hazard function $\lambda(t; x)$ with survival time T for an individual with possibly time-dependent covariate x , i.e.,

$$\lambda(t; x) = \lambda_0(t) \exp(\beta^\top x), \quad (12.17)$$

where $\lambda_0(t)$ is an arbitrary and unspecified base-line hazard function and $x^\top = (x_1, \dots, x_p)$ and $\beta^\top = (\beta_1, \dots, \beta_p)$. Cox generalized (12.17) this to a discrete logistic model expressing y as

$$\frac{\lambda(t; x)}{1 - \lambda(t; x)} = \frac{\lambda_0(t)}{1 - \lambda_0(t)} \exp(\beta^\top x). \quad (12.18)$$

Kamakura and Yanagimoto (1983) compared the estimators of regression parameters in the proportional hazards model (12.17) or (12.18) when we take the following methods; the Breslow-Peto (Breslow, 1974; Peto, 1972) method, the partial likeli-

hood (Cox, 1972, 1975) method and the generalized maximum likelihood method (Kalbfleish and Prentice, 1980; Miller, 1981).

The Score Test

12.3.1

In many applications it is necessary to test the significance of the estimated value, using for example the score test or the likelihood ratio test based on asymptotic results of large sample theory. First we express the three likelihood factors defined at each failure time as L_{BP}, L_{PL}, L_{GML} corresponding to the Breslow-Peto, the partial likelihood and the generalized maximum likelihood methods, respectively;

$$L_{BP}(\beta) = \frac{\prod_{i=1}^r \exp(\beta^\top x_i)}{\{\sum_{i=1}^n \exp(\beta^\top x_i)\}^r}, \tag{12.19}$$

$$L_{PL}(\beta) = \frac{\prod_{i=1}^r \exp(\beta^\top x_i)}{\sum_{\Psi} \prod_{i=1}^r \exp(\beta^\top x_{\psi_i})}, \tag{12.20}$$

$$L_{GML}(\beta) = \frac{\prod_{i=1}^r \lambda \exp(\beta^\top x_i)}{\prod_{i=1}^n \{1 + \lambda \exp(\beta^\top x_i)\}}, \tag{12.21}$$

where x_1, \dots, x_n denote covariate vectors for n individuals at risk at a failure time and x_1, \dots, x_r correspond to the failures, and Ψ denotes the set of all subsets $\{\psi_1, \dots, \psi_r\}$ of size r from $\{1, \dots, n\}$. The overall likelihood obtained by each method is the product of these cases of many failure times. It can be shown that the first derivatives of the three log likelihoods with respect β have the same values, i.e.,

$$\sum_{i=1}^r x_{ji} - \frac{r}{n} \sum_{i=1}^n x_{ji} \quad (j = 1, \dots, p)$$

at $\beta = 0$.

The Hessian matrices of the log likelihoods evaluated at $\beta = 0$ are respectively,

$$\begin{aligned} & -\left(\frac{r}{n}\right)S, \\ & -\left\{\frac{r(n-r)}{n(n-1)}\right\}S, \\ & -\left\{\frac{r(n-r)}{n^2}\right\}S, \end{aligned}$$

where S is a matrix whose elements s_{jk} are defined by

$$s_{jk} = \sum_{i=1}^n (x_{ji} - \bar{x}_j)(x_{ki} - \bar{x}_k).$$

The first two results were derived by Farewell and Prentice (1980). Maximizing out λ from L_{GML} gives the last one, which is obtained in an unpublished manuscript. Since

$$\frac{r}{n} \geq \frac{r(n-r)}{n(n-1)} > \frac{r(n-r)}{n^2},$$

we conclude that the Breslow–Peto approach is the most conservative one.

12.3.2 Evaluation of Estimators in the Cox Model

Farewell and Prentice (1980) pointed out in their simulation study that when the discrete logistic model is true the Breslow–Peto method causes downward bias compared to the partial likelihood method. This was proven in Kamakura and Yanagimoto (1983) for any sample when β is scalar-valued, i.e.,

2 **Theorem 2** (Kamakura and Yanagimoto, 1983)

Let $\widehat{\beta}_{BP}$ be the maximum likelihood estimator of $L_{BP}(\beta)$ and $\widehat{\beta}_{PL}$ be that of $L_{PL}(\beta)$. Suppose that all x_i 's are not identical. Then both $\widehat{\beta}_{BP}$ and $\widehat{\beta}_{PL}$ are unique, if they exist, and $\text{sgn}(\widehat{\beta}_{BP}) = \text{sgn}(\widehat{\beta}_{PL})$ and

$$|\widehat{\beta}_{BP}| \leq |\widehat{\beta}_{PL}|. \quad (12.22)$$

The equality in (12.22) holds when $\widehat{\beta}_{PL}$ is equal to zero or the number of ties r is equal to one.

1 **Corollary 1** (Kamakura and Yanagimoto, 1983)

The likelihood ratio test for $\beta = 0$ against $\beta \neq 0$ is also conservative if we use the Breslow–Peto method. The statement is also valid in the multivariate case.

This theorem and corollary confirm the conservatism of the Breslow–Peto approximation in relation to Cox's discrete model (Oaks, 2001).

12.3.3 Approximation of Partial Likelihood

Yanagimoto and Kamakura (1984) proposed an approximation method using full likelihood for the case of Cox's discrete model. Analytically the same problems appear in various fields of statistics. Prentice and Breslow (1978) and Farewell (1979) remarked that the inference procedure using the logistic model contains the same problems in case-control studies where data are summarized in multiple 2×2 or $k \times 2$ tables. The proportional hazards model provides a type of logistic model for the contingency table with ordered categories (Pregibon, 1982). As an extension of the proportional hazards model, the proportional intensity model in the point process is employed to describe an asthma attack in relation to environmental factors

(Korn and Whittemore, 1979; Yanagimoto and Kamakura, 1984). For convenience, although in some cases partial likelihood becomes conditional likelihood, we will use the term of partial likelihood.

It is worthwhile to explore the behavior of the maximum full likelihood estimator even when the maximum partial likelihood estimator is applicable. Both estimators obviously behave similarly in a rough sense, yet they are different in details. Identifying differences between the two estimators should be helpful in choosing one of the two.

We use the notation described in the previous section for expressing the two likelihoods. Differentiating $\log L_{PL}$ gives

$$LP(\beta) = \sum_{i=1}^r x_i - \frac{\sum_{\psi} \sum_{\psi} x_j \exp(\beta^T \sum_{\psi} x_j)}{\sum_{\psi} \exp(\beta^T \sum_{\psi} x_j)} = 0.$$

Differentiating $\log L_{GML}$ with respect to β and λ allows obtaining the maximum full likelihood estimator, i.e.,

$$\sum_{i=1}^r x_i - \sum_{i=1}^n \lambda x_i \frac{\exp(\beta^T x_i)}{1 + \lambda \exp(\beta^T x_i)} = 0$$

and

$$\frac{r}{\lambda} - \sum_{i=1}^n \frac{\exp(\beta^T x_i)}{1 + \lambda \exp(\beta^T x_i)}.$$

From the latter equation $\lambda(\beta)$ is uniquely determined for any fixed β . Using $\lambda(\beta)$, we define

$$LF(\beta) = \sum_{i=1}^r x_i - \sum_{i=1}^n \lambda(\beta) x_i \frac{\exp(\beta^T x_i)}{1 + \lambda \exp(\beta^T x_i)}.$$

The maximum full likelihood estimator, $\hat{\beta}_{GML}$, is a root of the equation $LF(\beta) = 0$. We denote $\lambda(\beta)$ by λ for simplicity.

Note that the entire likelihoods are the products over all distinct failure times T . Thus the likelihood equations in a strict sense are $\sum LP_t(\beta) = 0$ and $\sum LF_t(\beta) = 0$, where the summations extend over t in T . As far as we are concerned, the results in a single failure time can be straightforwardly extended to those with multiple failure times. Let us now focus on likelihood equations of a single failure time and suppress the suffix t .

Proposition 1 (Yanagimoto and Kamakura, 1984)

Let $K(\beta)$ be either of $LF(\beta)$ or $LP(\beta)$. Denote $\sum_{i=1}^n x_i/n$ by \bar{x} , and $x_{(1)} + \dots + x_{(r)}$ and $x_{(n-r+1)} + \dots + x_{(n)}$ by $L(x; r)$ and $U(x; r)$ respectively, where $x_{(1)}, \dots, x_{(n)}$ are ordered covariates in ascending order. $K(\beta)$ accordingly has the following four properties:

- (1) $K(0) = x_1 + \cdots + x_r - r\bar{x}$.
- (2) $K'(\beta)$ is negative for any β , that is, $K(\beta)$ is strictly decreasing.
- (3) $\lim_{\beta \rightarrow -\infty} K(\beta) = U(x; r)$.
- (4) $\lim_{\beta \rightarrow \infty} K(\beta) = L(x; r)$.

Extension to the case of vector parameter β is straightforward. From Proposition 1 it follows that if either of the two estimators exists, then the other also exists and they are uniquely determined. Furthermore, both the estimators have a common sign.

3 Theorem 3 (Yanagimoto and Kamakura, 1984)

Suppose that $\sum(x_i - \bar{x})^2 \neq 0$. The functions $LP(\beta)$ and $LF(\beta)$ then have a unique intersection at $\beta = 0$. It also holds that $LP(\beta) < LF(\beta)$ for $\beta > 0$. The reverse inequality is valid for $\beta < 0$.

The above theorem proves that $\hat{\beta}_{GML} > \hat{\beta}_{PL}$ for the case of $LP(0) = LF(0) > 0$.

To quantitatively compare the behaviors of $LF(\beta)$ and $LP(\beta)$, their power expansions are presented near the origin. Since both functions behave similarly, it is expected that the quantitative difference near the origin is critical over a wide range of β . Behavior near the origin is of practical importance for studying the estimator and test procedure.

2 Proposition 2 (Yanagimoto and Kamakura, 1984)

The power expansions of $LF(\beta)$ and $LP(\beta)$ near the origin up to the third order are as follows: for $n \geq 4$,

(1)

$$LF(\beta) \approx \sum_{i=1}^r x_i - \left[r\bar{x} + \frac{r(n-r)}{n^2} s_2 \beta + \frac{1}{2} \frac{r(n-r)(n-2r)}{n^3} s_3 \beta^2 + \frac{1}{6} \frac{r(n-r)}{n^5} \{n(n^2 - 6rn + 6r^2) s_4 - 3(n-2r)^2 s_2^2\} \beta^3 \right],$$

(2) (Cox, 1970)

$$LP(\beta) \approx \sum_{i=1}^r x_i - \left[r\bar{x} + \frac{r(n-r)}{n(n-1)} s_2 \beta + \frac{1}{2} \frac{r(n-r)(n-2r)}{n(n-1)(n-2)} s_3 \beta^2 + \frac{1}{6} \frac{r(n-r)}{n^2(n-1)(n-2)(n-3)} \{n(n^2 - 6rn + 6r^2 + n) s_4 + 3(r-1)n(n-r-1) s_2^2\} \beta^3 \right],$$

where $s_k = \sum(x_i - \bar{x})^k$, $k = 2, 3$ and 4 .

The function $LF(\beta)$ has a steeper slope near the origin than $LP(\beta)$. The relative ratio is $n/(n-1)$, which indicates that $LF(n\beta/(n-1))$ is close to $LP(\beta)$ near the origin. The power expansion of $LA(\beta) = LF(n\beta/(n-1))$ is expressed by

$$LA(\beta) \approx \sum_{i=1}^r x_i - \left\{ r\bar{x} + \frac{r(n-r)}{n(n-1)} s_2\beta + \left(\frac{n}{n-1}\right)^2 c_3\beta^2 + \left(\frac{n}{n-1}\right)^3 c_4\beta^3 \right\}, \quad (12.23)$$

where c_3 and c_4 are coefficients of order 2 and 3 of $LF(\beta)$. Although $LA(\beta)$ is defined to adjust the coefficient of $LF(\beta)$ of order 1 to that of $LP(\beta)$, the coefficient of order 2 of $LA(\beta)$ becomes closer to that of $LP(\beta)$ than that of $LF(\beta)$. The following approximations are finally obtained.

$$LP(\beta) \approx LA(\beta), \quad (12.24)$$

$$\hat{\beta}_{PL} \approx \frac{(n-1)\hat{\beta}_{GML}}{n}. \quad (12.25)$$

The proposed approximated estimator and test statistic are quite helpful in cases of multiple 2×2 table when the value of both n and r are large (Yanagimoto and Kamakura, 1984).

Multiple Failures and Counting Processes

12.4

The standard methods of survival analysis can be generalized to include multiple failures simply defined as a series of well-defined event occurrences. For example, in software reliability, engineers are often interested in detecting software bugs. Inference from a single counting process has been studied in detail (Cox and Lewis, 1966; Musa et al., 1987), with multiple independent processes being considered as a means to estimate a common cumulative mean function from a nonparametric or semi-parametric viewpoint (Lawless and Nadeau, 1993; Nelson, 1992). Kamakura (1996) discussed problems associated with parametric conditional inference in models with a common trend parameter or possibly different base-line intensity parameters.

Intensity Function

12.4.1

For multiple failures, intensity functions correspond to hazard functions in that the intensity function is defined as discussed next.

In time interval $[t_0, t]$ we define the number of occurrences of events or failures as $N(t)$. The Poisson counting process $\{N(t) : t \geq t_0\}$ is given such that it satisfies the following three conditions for $t \geq t_0$.

1. $\Pr\{N(t_0) = 0\} = 1$
2. The increment $N_{s,t} = N(t) - N(s)$ ($t_0 \geq s < t$) has a Poisson distribution with the mean parameter $\Lambda_t - \Lambda_s$, for some positive and increasing function in t .

3. $\{N_t : t \geq t_0\}$ is a process of independent increments. That is, for any $(t_0 < t_1 < t_2 < \dots < t_n, n$ increments, $N(t_1) - N(t_0), \dots, N(t_n) - N(t_{n-1})$ are mutually independent.

For this counting process $\{N(t) : t \geq t_0\}$ we can define the intensity function as

$$\lambda(t) = \lim_{\Delta \rightarrow 0} \frac{1}{\Delta} \Pr\{N(t + \Delta t) - N(t) = 1 | H(t)\}, \quad (12.26)$$

where $H(t)$ is the history of the process up to t :

$$H(t) = \{N(u) : t_0 \leq u \leq t\}.$$

Note that

$$\Lambda(t) = \int_{t_0}^t \lambda(t) dt.$$

Expectation of $E[N_{s,t}]$ becomes

$$E[N_{s,t}] = \sum_{n=0}^{\infty} n \Pr\{N_{n,s} = n\} = \Lambda_t - \Lambda_s, \quad (12.27)$$

and

$$\lambda(t) = \frac{d}{dt} \Lambda_t = \frac{d}{dt} E[N(t)]. \quad (12.28)$$

The nonparametric estimate of the intensity function is easy to determine and is quite useful for observing the trend of a series of events. If a data set of failure times $\{t_1, t_2, \dots, t_n\}$ is available, assuming constant intensity in $(t_{k-1}, t_k]$, then

$$\lambda(t) = \lambda_k \quad (t_{k-1} < t \leq t_k),$$

and the nonparametric ML estimates becomes

$$\lambda_k = \frac{1}{t_k - t_{k-1}} \quad (k = 1, \dots, n), \quad (12.29)$$

where $t_0 = 0$.

12.4.2 Multiple Counting Processes

We assume several independent counting processes $\{N_k(t_k), \text{ i.e., } 0 < t_k \leq \tau_k, k = 1, \dots, K\}$. The cumulative mean function for $N_k(t)$ is expressed by

$$M_k(t) = E\{N_k(t)\}. \quad (12.30)$$

Nelson (1992) described a method for estimating the cumulative mean function of an identically distributed process without assuming any Poisson process

structure, while Lawless and Nadeau (1993) developed robust variance estimates based on the Poisson process. All these methods are basically concerned with nonparametric estimation. Here, parametric models for effectively acquiring information on the trend of an event occurrence are dealt with. Kamakura (1996) considered generalized versions of two primal parametric models to multiple independent counting processes under the framework of a nonhomogeneous Poisson process.

Cox and Lewis (Cox and Lewis, 1966) considered a log-linear model for trend testing a single counting process, i.e.,

$$\lambda(t) = \exp(\alpha + \beta t), \tag{12.31}$$

where $\lambda(t)$ is the intensity function corresponding to the derivative of the mean function in the continuous case. Note that for a single case the subscript k is omitted. They assumed the above nonhomogeneous Poisson process and gave a simple test statistic for $H_0 : \beta = 0$ against $H_A : \beta \neq 0$, i.e.,

$$U = \frac{\sum_{i=1}^n t_i - \frac{1}{2}t_0}{\tau_0 \sqrt{\frac{n}{12}}}. \tag{12.32}$$

The distribution of this statistic steeply converges to the standard normal distribution when $n \rightarrow \infty$. This statistic is sometimes called the U statistic and is frequently applied to trend testing in reliability engineering.

Kamakura (1996) generalized this log-linear model to the multiple case, with the log-linear model for k -th individual being

$$\lambda_k(t) = \exp(\alpha_k + \beta t). \tag{12.33}$$

In this modeling we assume the common trend parameter β and are mainly interested in estimating and testing this parameter. The full likelihood for the model becomes

$$\begin{aligned} L(\beta, \alpha_1, \alpha_2, \dots, \alpha_K) &= \prod_{k=1}^K \left[\left\{ \prod_{i=1}^{n_k} \lambda_k(t_{ki}) \right\} \exp \left\{ - \int_0^{\tau_k} \lambda_k(u) du \right\} \right] \\ &= \exp \left\{ \sum_{k=1}^K n_k \alpha_k + \beta \sum_{k=1}^K \sum_{i=1}^{n_k} t_{ki} - \frac{1}{\beta} \sum_{k=1}^K e^{\alpha_k} (e^{\beta \tau_k} - 1) \right\}. \end{aligned} \tag{12.34}$$

If K is large, it is difficult to compute all parameter estimates based on such full likelihood.

Given $N_k(\tau_k) = n_k, k = 1, 2, \dots, K$, conditional likelihood is considered as

$$CL(\beta | N_k(\tau_k) = n_k, i = 1, \dots, K) = \frac{\prod_{k=1}^K (n_k!) \beta^{\sum n_k} e^{\beta \sum \sum t_{ki}}}{\prod_{k=1}^K (e^{\beta \tau_k} - 1)^{n_k}}. \tag{12.35}$$

Note that the nuisance parameter α_k 's do not appear. Fisher information is calcu-

lated as

$$I(\beta) = E \left[-\frac{\partial^2 \log CL}{\partial \beta^2} \right] \\ = \begin{cases} \sum_{k=1}^K n_k \left\{ \frac{1}{\beta^2} - \frac{\tau_k^2 e^{-\beta \tau_k}}{(1 - e^{-\beta \tau_k})^2} \right\} & (\beta \neq 0) \\ \frac{1}{12} \sum_{k=1}^K n_k \tau_k^2 & (\beta = 0) \end{cases} . \quad (12.36)$$

The test statistic obtained from the above calculations becomes

$$U_k = \frac{\log CL|_{\beta=0}}{\sqrt{I(0)}} \\ = \frac{\sum_{k=1}^K \sum_{i=1}^{n_k} t_{ki} - \frac{1}{2} \sum_{k=1}^K n_k \tau_k}{\sqrt{\frac{1}{12} \sum_{k=1}^K n_k \tau_k^2}} . \quad (12.37)$$

To obtain the conditional estimate, numerical calculations are required such as Newton–Raphson method. However, the log conditional likelihood and its derivatives are not computable at the origin of the parameter β . In such a case, Taylor series expansions of the log conditional likelihood are used around the origin (Kamakura, 1996).

12.4.3 Power Law Model

Crow (1982) considered the power law model, sometimes called the Weibull process model. This model was generalized to the multiple case using the following intensity for the k -th individual (Kamakura, 1996):

$$\lambda_k(t) = \theta_k m t^{m-1} . \quad (12.38)$$

In this case it is easy to calculate the MLE. Direct calculation of the likelihood gives rise to the MLE \hat{m} and $\hat{\theta}_k$ i.e.,

$$\hat{m} = \frac{\sum_{k=1}^K n_k}{\sum_{k=1}^K \sum_{i=1}^{n_k} \log \left(\frac{t_k}{t_{ki}} \right)} , \quad (12.39)$$

$$\hat{\theta}_k = \frac{n_k}{t_k^{\hat{m}}} . \quad (12.40)$$

Putting

$$Z = \frac{2m \sum_{k=1}^K n_k}{\hat{m}} , \quad (12.41)$$

the distribution of Z becomes a chi-square with $2 \sum_{k=1}^K n_k$ degrees of freedom. Based on this result we can make an inference of the common parameter m .

Models Suitable for Conditional Estimation

12.4.4

Estimation based on conditional likelihood allows effectively eliminating the nuisance parameter and obtaining information on the structure parameter. Let us now consider the class of nonhomogeneous Poisson process models which are specified by the intensity parameterized by two parameters. The first parameter α is concerned with the base line occurrences for the individual, while the second parameter β is concerned with the trend of intensity. For simplicity, the property of the intensity for $K = 1$ is examined. Using conditional likelihood is convenient because the nuisance parameter α need not be known. This is of great importance in multiple intensity modeling, i.e.,

Theorem 4 (Kamakura, 1996)

4

Conditional likelihood does not include the nuisance parameter α iff the intensity is factorized as two factors, a function of α and a function of β and the time t , in the class of nonhomogeneous Poisson process models. That is, the intensity is expressed as

$$\lambda(t; \alpha, \beta) = h(\alpha)g(\beta; t), \quad a.s. \quad (12.42)$$

Several intensity models for software reliability are described in Musa et al. (1987): the log-linear model, geometric model, inverse linear model, inverse polynomial model, and power law model, all of which are included in this class satisfying the condition of the theorem.

Acknowledgements. This work has been partially supported financially by Chuo University as one of the 2003 Research Projects for Promotion of Advanced Research at Graduate School.

References

- Breslow, N. E. (1974). Covariance analysis of censored survival data. *Biometrics*, 30:89–99.
- Castillo, E. and Hadi, A. S. (1995). Modeling lifetime data with application to fatigue models. *Journal of American Statistical Association*, 90:1041–1054.
- Cohen, A. C. (1965). Maximum likelihood estimation in the Weibull distribution based on complete and censored samples. *Technometrics*, 5:579–588.
- Cohen, A. C. and Whitten, B. J. (1988). *Parameter estimation in reliability and life span models*. Marcel Dekker, New York.

- Cox, D. R. (1970). *The analysis of binary data*. Methuen, London.
- Cox, D. R. (1972). Regression models and life tables (with discussion). *Journal of Royal Statistical Society: B*, 34:187–220.
- Cox, D. R. (1975). Partial likelihood. *Biometrika*, 62:269–276.
- Cox, D. R. and Lewis, P. A. W. (1966). *The statistical analysis of series of events*. Methuen, London.
- Crow, L. H. (1982). Confidence interval procedures for the Weibull process with applications to reliability growth. *Technometrics*, 24:67–72.
- Dubey, S. D. (1967). Normal and Weibull distributions. *Naval Research Logistics Quarterly*, 14:69–79.
- Farewell, V. T. (1979). Some results on the estimation of logistic models based on retrospective data. *Biometrika*, 66:27–32.
- Farewell, V. T. and Prentice, R. L. (1980). The approximation of partial likelihood with emphasis on case-control studies. *Biometrika*, 67:273–278.
- Greenwood, M. (1926). *A report on the natural duration of cancer: Appendix I The 'Errors of Sampling' of the survivorshi tables: Reports on Public Health and Medical Subjects*. His Majesty's Stationery Office, London.
- Johnson, N. L., Kotz, S., and Balakrishna, N. (1994). *Continuous Univariate Distributions*. John Wiley, New York.
- Kalbfleish, J. D. and Prentice, R. L. (1980). *The statistical analysis of failure time data*. John Wiley, New York.
- Kamakura, T. (1996). Trend analysis of multiple counting processes. In Jewell, N. P., Kimber, A. C., Lee, M.-L. T., and Whitmore, G. A., editors, *Lifetime Data: Models in Reliability and Survival Analysis*, pages 149–156. Kluwer Academic Publishers, Dordrecht.
- Kamakura, T. and Yanagimoto, T. (1983). Evaluation of the regression parameter estimators in the proportional hazard model. *Biometrika*, 70:530–533.
- Kaplan, E. L. and Meier, P. (1958). Nonparametric estimation from incomplete observations. *Journal of American Statistical Association*, 53:457–481.
- Korn, E. L. and Whittemore, A. (1979). Methods for analysing panel studies of acute health effects of air pollution. *Biometrics*, 35:795–802.
- Lawless, J. F. and Nadeau, J. C. (1993). Some simple robust methods for the analysis of recurrent events. University of Waterloo IIQP Research Report.
- Menon, M. (1963). Estimation of the shape and scale parameters of the Weibull distributions. *Technometrics*, 5:175–182.
- Miller, R. G. (1981). *Survival analysis*. John Wiley, New York.
- Musa, J. D., Iannino, A., and Okumoto, K. (1987). *Software Reliability*. McGraw-Hill, New York.
- Nagatsuka, H. and Kamakura, T. (2003). A new method of inference for Weibull shape parameter (in Japanese). *Journal of Reliability Engineering Association of Japan*, 25:583–593.
- Nagatsuka, H. and Kamakura, T. (2004). Parameter estimation of the shape parameter of the Castillo–Hadi model. *Communications in Statistics: Theory and Methods*, 33:15–27.

- Nelson, W. B. (1992). Confidence limits for recurrence data – applied to cost or number of product repairs and of disease episodes. *Technometrics*, 22:1023–1031.
- Oaks, D. (2001). Biometrika centenary: Survival analysis. *Biometrika*, 88:99–142.
- Peto, R. (1972). Discussion of paper by D. R. Cox. *Journal of Royal Statistical Society: B*, 34:205–207.
- Pregibon, D. (1982). Resistant fits for some commonly used logistic models with medical applications. *Biometrics*, 38:485–498.
- Prentice, R. L. and Breslow, N. E. (1978). Restrospective studies and failure time models. *Biometrika*, 65:153–158.
- Yanagimoto, T. and Kamakura, T. (1984). The maximum full and partial likelihood estimators in the proportional hazard model. *Annals of the Institute of Statistical Mathematics*, 36:363–373.

Data and Knowledge Mining

III.13

Adalbert Wilhelm

13.1	<i>Data Dredging and Knowledge Discovery</i>	789
13.2	<i>Knowledge Discovery in Databases</i>	790
13.3	<i>Supervised and Unsupervised Learning</i>	792
13.4	<i>Data Mining Tasks</i>	793
	Description and Summarization	793
	Descriptive Modeling.....	794
	Predictive Modeling.....	795
	Discovering Patterns and Rules	795
	Retrieving Similar Objects	796
13.5	<i>Data Mining Computational Methods</i>	797
	Numerical Data Mining	797
	Visual Data Mining	803

Data Dredging and Knowledge Discovery

Data mining was one of the buzz-words at the verge of the third millennium. It was already a multi-million dollar industry in the late 1990s and experts expected a continuing growth for the first decade of the 21st century. Although this expectation has not quite materialized in recent years, data mining still is an important field of scientific research with great potential for commercial usage. The ubiquitous computer makes it possible to collect huge data bases that contain potentially valuable information. Sophisticated analysis techniques are needed to explore these large, often heterogeneous, data sets and to extract the small pieces of information that are valuable to the data owner.

The importance of exploring and analyzing real data sets is not new to statistics. It has been reinforced in the late 1960s by John W. Tukey who realized that putting too much emphasis on the mathematical theories of statistics did not help in solving the real world problems. It was his mantra that statistical work is detective work (Tukey, 1969) and that one should let the data speak for itself. The branch of exploratory data analysis emerged, but was dismissed by mathematical statisticians for a long period of time. Many of them proclaimed that proper statistical analysis must be based on hypothesis and distributional assumptions. Their argument was that looking at data before formulating a scientific hypothesis will bias the hypothesis towards what the data might show. The term data mining typically was used in a derogatory connotation. The argument culminated in the reproach of improper scientific use, the reproach of torturing the data until it confesses everything.

The advent of information technology that allowed to easily collect and store data of previously unimaginable quantities brought a rapid change to the scene and superseded academic disputes. Once the computer power and technology was there, that made it easy to collect information of all customers in a super market, or for all customers of a telephone company, the need arose to make use of these large information sources.

Now, Data Mining is a thriving field of research and application, to which both statisticians and computer scientists have contributed new ideas and new techniques. In this contribution, we will introduce the main components, tasks, and computational methods for data mining. After an attempt to define data mining, we relate it to the larger field of knowledge discovery in databases in Sect. 13.2. Section 13.3 deals with the two flavors of learning from data: supervised and unsupervised learning. We will then discuss the different data mining tasks in Sect. 13.4, before we present the computational methods to tackle them in Sect. 13.5. In the final Sect. 13.5.2, we present some recent trends in visual methods for data mining.

Knowledge Discovery in Databases

There are almost as many differing definitions of the term “Data Mining” as there are authors who have written about it. Since Data Mining sits at the interface of a variety of fields, e.g. computer science, statistics, artificial intelligence, business information systems, and machine learning, its definition changes with the field’s perspective. Computer scientists, typically, refer to Data Mining as a clearly defined part of the Knowledge Discovery in Databases (KDD) process, while many statisticians use Data Mining as a synonym for the whole KDD process.

To get a flavor of both the variation as well as the common core of data and knowledge mining, we cite some of the definitions used in the literature.

KDD is the non-trivial process of identifying valid, novel, potentially useful, and ultimately understandable patterns in data.

(Fayyad et al., 1996)

Knowledge discovery is a knowledge-intensive task consisting of complex interactions, protracted over time, between a human and a (large) database, possibly supported by a heterogenous suite of tools.

(Brachman and Anand, 1996)

[Data Mining is] a step in the KDD process consisting of particular data mining algorithms that, under some acceptable computational efficiency limitations, produce a particular enumeration of patterns.

(Fayyad et al., 1996)

[Data Mining is] a folklore term which indicates application, under human control, of low-level data mining methods. Large scale automated search and interpretation of discovered regularities belong to KDD, but are typically not considered part of data mining.

(Kloesgen and Zytkow, 1996)

[Data Mining is] used to discover patterns and relationships in data, with an emphasis on large observational data bases. It sits at the common frontiers of several fields including Data Base Management, Artificial Intelligence, Machine Learning, Pattern Recognition, and Data Visualization.

(Friedman, 1998)

[Data Mining is] the process of secondary analysis of large databases aimed at finding unsuspected relationships which are of interest or value to the database owners.

(Hand, 1998)

Data Mining is the analysis of (often large) observational data sets to find unsuspected relationships and to summarize the data in novel ways that are both understandable and useful to the data owner.
(Hand et al., 2001)

From these definitions the essence is that we are talking about exploratory analysis of large data sets. Two further aspects are the use of computer-based methods and the notion of secondary and observational data. The latter means that the data do not come from experimental studies and that data was originally collected for some other purpose, either for a study with different goals or for record-keeping reasons. These four characteristics in combination distinguish the field of Data Mining from traditional statistics. The exploratory approach in Data Mining clearly defines the goal of finding patterns and generating hypothesis, which might later on be subject of designed experiments and statistical tests. Data sets can be large at least in two different aspects. The most common one is in form of a large number of observations (cases). Real world applications usually are also large in respect of the number of variables (dimensions) that are represented in the data set. Data Mining is also concerned with this side of largeness. Especially in the field of bioinformatics, many data sets comprise only a small number of cases but a large number of variables. Secondary analysis implies that the data can rarely be regarded as a random sample from the population of interest and may have quite large selection biases. The primary focus in investigating large data sets tends not to be the standard statistical approach of inferencing from a small sample to a large universe, but more likely partitioning the large sample into homogeneous subsets.

The ultimate goal of Data Mining methods is not to find patterns and relationships as such, but the focus is on extracting knowledge, on making the patterns understandable and usable for decision purposes. Thus, Data Mining is the component in the KDD process that is mainly concerned with extracting patterns, while Knowledge Mining involves evaluating and interpreting these patterns. This requires at least that patterns found with Data Mining techniques can be described in a way that is meaningful to the data base owner. In many instances, this description is not enough, instead a sophisticated model of the data has to be constructed.

Data pre-processing and data cleansing is an essential part in the Data and Knowledge Mining process. Since data mining means taking data from different sources, collected at different time points, and at different places, integration of such data as input for data mining algorithms is an easily recognized task, but not easily done. Moreover, there will be missing values, changing scales of measurement, as well as outlying and erroneous observations. To assess the data quality is a first and important step in any scientific investigation. Simple tables and statistical graphics give a quick and concise overview on the data, to spot data errors and inconsistencies as well as to confirm already known features. Besides the detection of uni- or bivariate outliers graphics and simple statistics help in assessing the quality of the data in general and to summarize the general behavior.

It is worth noting that many organizations still report that as much as 80% of their effort for Data and Knowledge Mining goes into supporting the data cleansing and transformation process.

Supervised and Unsupervised Learning

13.3

Data and Knowledge Mining is learning from data. In this context, data are allowed to speak for themselves and no prior assumptions are made. This learning from data comes in two flavors: supervised learning and unsupervised learning. In supervised learning (often also called directed data mining) the variables under investigation can be split into two groups: explanatory variables and one (or more) dependent variables. The target of the analysis is to specify a relationship between the explanatory variables and the dependent variable as it is done in regression analysis. To apply directed data mining techniques the values of the dependent variable must be known for a sufficiently large part of the data set.

Unsupervised learning is closer to the exploratory spirit of Data Mining as stressed in the definitions given above. In unsupervised learning situations all variables are treated in the same way, there is no distinction between explanatory and dependent variables. However, in contrast to the name undirected data mining there is still some target to achieve. This target might be as general as data reduction or more specific like clustering. The dividing line between supervised learning and unsupervised learning is the same that distinguishes discriminant analysis from cluster analysis. Supervised learning requires that the target variable is well defined and that a sufficient number of its values are given. For unsupervised learning typically either the target variable is unknown or has only been recorded for too small a number of cases.

The large amount of data that is usually present in Data Mining tasks allows to split the data file in three groups: training cases, validation cases and test cases. Training cases are used to build a model and estimate the necessary parameters. The validation data helps to see whether the model obtained with one chosen sample may be generalizable to other data. In particular, it helps avoiding the phenomenon of overfitting. Iterative methods incline to result in models that try to do too well. The data at hand is perfectly described, but generalization to other data yields unsatisfactory outcomes. Not only different estimates might yield different models, usually different statistical methods or techniques are available for a certain statistical task and the choice of a method is open to the user. Test data can be used to assess the various methods and to pick the one that does the best job on the long run.

Although we are dealing with large data sets and typically have abundant cases, partially missing values and other data peculiarities can make data a scarce resource and it might not be easily achievable to split the data into as many subsets as there are necessary. Resampling and cross-validation techniques are often used in combination to data and computer intensive methods in Data Mining.

13.4 Data Mining Tasks

The cycle of data and knowledge mining comprises various analysis steps, each step focusing on a different aspect or task. Hand et al. (2001) propose the following categorization of data mining tasks.

13.4.1 Description and Summarization

At the beginning of each data analysis is the wish and the need to get an overview on the data, to see general trends as well as extreme values rather quickly. It is important to familiarize with the data, to get an idea what the data might be able to tell you, where limitations will be, and which further analyses steps might be suitable. Typically, getting the overview will at the same time point the analyst towards particular features, data quality problems, and additional required background information. Summary tables, simple univariate descriptive statistics, and simple graphics are extremely valuable tools to achieve this task.

Unwin et al. (2002) report from a study of 50,000 car insurance policies during which the following difficulties emerged amongst others (see Fig. 13.1).

- (a) Barcharts of the categorical variables revealed that several had too many categories. Sex had seven, of which four were so rare as to presumably be unknowns or errors of some kind. The third large category turned out to be very reasonable: if a car was insured by a firm, the variable sex was coded as “firm”. This had not been explained in advance and was obviously useful for a better grasp of the data.
- (b) A histogram of date of birth showed missing values, a fairly large number (though small percentage) of underage insured persons, and a largish number born in 1900, who had perhaps been originally coded as “o” or “oo” for unknown. Any analytic method using such a variable could have given misleading results.
- (c) Linking the barchart of gender from (a) and the histogram of age from (b) showed quite plausibly that many firms had date of birth coded as missing, but not all. This led to further informative discussions with the data set owners.

Checking data quality is by no means a negative part of the process. It leads to deeper understanding of the data and to more discussions with the data set owners. Discussions lead to more information about the data and the goals of the study.

Speed of the data processing is an important issue at this step. For simple tasks – and data summary and description are typically considered to be simple tasks, although it is generally not true – users are not willing to spend much time. A frequency table or a scatterplot must be visible in the fraction of a second, even when it comprises a million observations. Only some computer programs are able to achieve this. Another point is a fast scan through all the variables: if a program requires an explicit and lengthy specification of the graph or table to

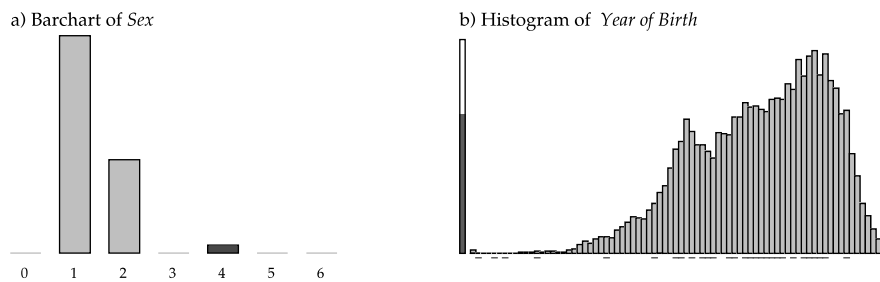


Figure 13.1. Linked highlighting reveals structure in the data and explains unusual results of one variable quite reasonably. Barchart of Sex of car insurance policy holders on the *left*, Histogram of year of birth of policy holders on the *right*. Highlighted are cases with *Sex* = 4 (firm). The *lines* under some of the bins in the histogram indicate small counts of highlighted cases that can't be displayed proportionally

be created, a user typically will end this tedious endeavor after a few instances. Generic functions with context-sensitive and variable-type-dependent responses provide a viable solution to this task. On the level of standard statistical data sets this is provided by software like XploRe, S-Plus and R with their generic functions *summary* and *plot*. Generic functions of this kind can be enhanced by a flexible and interactive user environment which allows to navigate through the mass of data, to extract the variables that show interesting information on the first glance and that call for further investigation. Currently, no system comes close to meet these demands, future systems hopefully will do.

Descriptive Modeling

13.4.2

General descriptions and summaries are an important starting point but more exploration of the data is usually desired. While the tasks in the previous section have been guided by the goal of summary and data reduction, descriptive modeling tries to find models for the data. In contrast to the subsequent section, the aim of these models is to describe, not to predict models. As a consequence, descriptive models are used in the setting of unsupervised learning. Typical methods of descriptive modeling are density estimation, smoothing, data segmentation, and clustering. There are by now some classics in the literature on density estimation (Scott, 1992) and smoothing (Härdle, 1991). Clustering is a well-studied and well-known technique in statistics. Many different approaches and algorithms, distance measures and clustering schemes have been proposed. With large data sets all hierarchical methods have extreme difficulties with performance. The most widely used method of choice is *k*-means clustering. Although *k*-means is not particularly tailored for a large number of observations, it is currently the only clustering scheme that has gained positive reputation in both the computer science and the statistics community. The reasoning behind cluster analysis is the assumption that the data set contains natural clusters which, when discovered, can be characterized and

labeled. While for some cases it might be difficult to decide to which group they belong, we assume that the resulting groups are clear-cut and carry an intrinsic meaning. In segmentation analysis, in contrast, the user typically sets the number of groups in advance and tries to partition all cases in homogeneous subgroups.

13.4.3 Predictive Modeling

Predictive modeling falls into the category of supervised learning, hence, one variable is clearly labeled as target variable Y and will be explained as a function of the other variables X . The nature of the target variable determines the type of model: classification model, if Y is a discrete variable, or regression model, if it is a continuous one. Many models are typically built to predict the behavior of new cases and to extend the knowledge to objects that are new or not yet as widely understood. Predicting the value of the stock market, the outcome of the next governmental election, or the health status of a person Banks use classification schemes to group their costumers into different categories of risk.

Classification models follow one of three different approaches: the discriminative approach, the regression approach, or the class-conditional approach. The discriminative approach aims in directly mapping the explanatory variables X to one of the k possible target categories y_1, \dots, y_k . The input space X is hence partitioned into different regions which have a unique class label assigned. Neural networks and support vector machines are examples for this. The regression approach (e.g. logistic regression) calculates the posterior class distribution $P(Y | x)$ for each case and chooses the class for which the maximum probability is reached. Decision trees (CART, C5.0, CHAID) classify for both the discriminative approach and the regression approach, because typically the posterior class probabilities at each leaf are calculated as well as the predicted class. The class-conditional approach starts with specifying the class-conditional distributions $P(X | y_i, \theta_i)$ explicitly. After estimating the marginal distribution $P(Y)$, Bayes rule is used to derive the conditional distribution $P(Y | x)$. The name Bayesian classifiers is widely used for this approach, erroneously pointing to a Bayesian approach versus a frequentist approach. Mostly, plug-in estimates $\hat{\theta}_i$ are derived via maximum likelihood. The class-conditional approach is particularly attractive, because they allow for general forms of the class-conditional distributions. Parametric, semi-parametric, and non-parametric methods can be used to estimate the class-conditional distribution. The class-conditional approach is the most complex modeling technique for classification. The regression approach requires fewer parameters to fit, but still more than a discriminative model. There is no general rule which approach works best, it is mainly a question of the goal of the researcher whether posterior probabilities are useful, e.g. to see how likely the “second best” class would be.

13.4.4 Discovering Patterns and Rules

The realm of the previous tasks has been much within the statistical tradition in describing functional relationships between explanatory variables and tar-

get variables. There are situations where such a functional relationship is either not appropriate or too hard to achieve in a meaningful way. Nevertheless, there might be a pattern in the sense that certain items, values or measurements occur frequently together. Association Rules are a method originating from market basket analysis to elicit patterns of common behavior. Let us consider an example originating from data that is available as one of the example data files for the SAS Enterprise Miner. For this data (in the following referred to as the SAS Assocs Data) the output for an association query with $\text{minconf} = 50$ and $\text{minsup} = 3$, limited by a maximum of 4 items per rule generated by the SAS Enterprise Miner consists of 1000 lines of the form shown in Table 13.1.

Table 13.1. Examples of association rules as found in the SAS Assocs Data by the SAS Enterprise Miner Software. 1000 rules have been generated: 47 including 2 items, 394 with 3 items and 559 with 4 items

# items	conf	supp	count	Rule
2	82.62	25.17	252	artichok → heineken
2	78.93	25.07	251	soda → cracker
2	78.09	22.08	221	turkey → olives
...				
3	95.16	5.89	59	soda & artichok → heineken
3	94.31	19.88	199	avocado & artichok → heineken
3	93.23	23.38	234	soda & cracker → heineken
...				
4	100.00	3.1	31	ham & corned beef & apples → olives
4	100.00	3.1	31	ham & corned beef & apples → hering
4	100.00	3.8	38	steak & soda & heineken → cracker
...				

The practical use of association rules is not restricted to finding the general trend and the norm behavior, association rules have also been used successfully for detecting unusual behavior in fraud detection.

Retrieving Similar Objects

13.4.5

The world wide web contains an enormous amount of information in electronic journal articles, electronic catalogs, and private and commercial homepages. Having found an interesting article or picture, it is a common desire to find similar objects quickly. Based on key words and indexed meta-information search engines are providing us with this desired information. They can not only work on text documents, but to a certain extent also on images. Semi-automated picture retrieval combines the ability of the human vision system with the search capacities of the computer to find similar images in a data base.

Data Mining Computational Methods

When talking about Data Mining methods people tend to refer to numerical methods mostly. However, there are many reasons to consider visual approaches to data mining as equally important. We start with a concise discussion of numerical data mining methods and expand a little bit more on visual data mining later.

Numerical Data Mining

Decision Trees

Decision trees are popular and powerful methods to classify cases as well as to predict values. Their attractiveness is mainly due to the fact that the basic principle of tree-based methods as well as their outcome are easy to understand. The basic principle is the hierarchical division of all observations into subcategories in such a way that the resulting subcategories differ from each other as much as possible while the subcategories itself are as homogenous as possible. The outcome of a decision tree is a rule that can easily be expressed in plain English and as easily in any data base query language to quickly and repeatedly apply it to new data.

Decision trees are a supervised learning technique and require that we can extract a target variable. Depending on the scale of the target variable two types of decision trees are distinguished: classification trees, if the dependent variable is categorical, and regression trees, if the dependent variable is continuous. A full analysis with decision trees comprises two steps, growing a tree and pruning a tree.

Let us briefly describe the basic principle of growing a binary tree. Given observations for n cases on the dependent variable Y and p explanatory variables X_1, \dots, X_p we first assign all cases into the root node of the tree. The typical algorithms are greedy algorithms that enumerate all possible splits to find the best split for the current node under investigation. Thus, for each explanatory variable X_i all possible splits between values are examined. During this examination the cases in the node are partitioned into two groups (for binary trees) and a diversity measure is calculated for each potential subnode. Commonly used diversity measures are deviance, entropy, misclassification rate, and the Gini coefficient. The one split that yields the largest information gain will be taken. Information gain is measured here in terms of the decrease of the diversity measure when going from the parent node to the subnode. The algorithm uses the same process recursively to build a decision tree for each subnode. A node will not be further split if it is pure or if one of the subnodes would contain too few cases. There are mainly three families of tree growing algorithms:

- The CART family (CART, IND CART, Splus CART, etc.)
- The ML family (ID3, C4.5, C5 and other derivatives, etc.)
- The AID family (THAID, CHAID, XAID, TREEDISC, etc.)

These families mainly differ in the type of splitting criterion they use and in the background of their origin. The AID family grew out of the social sciences and uses

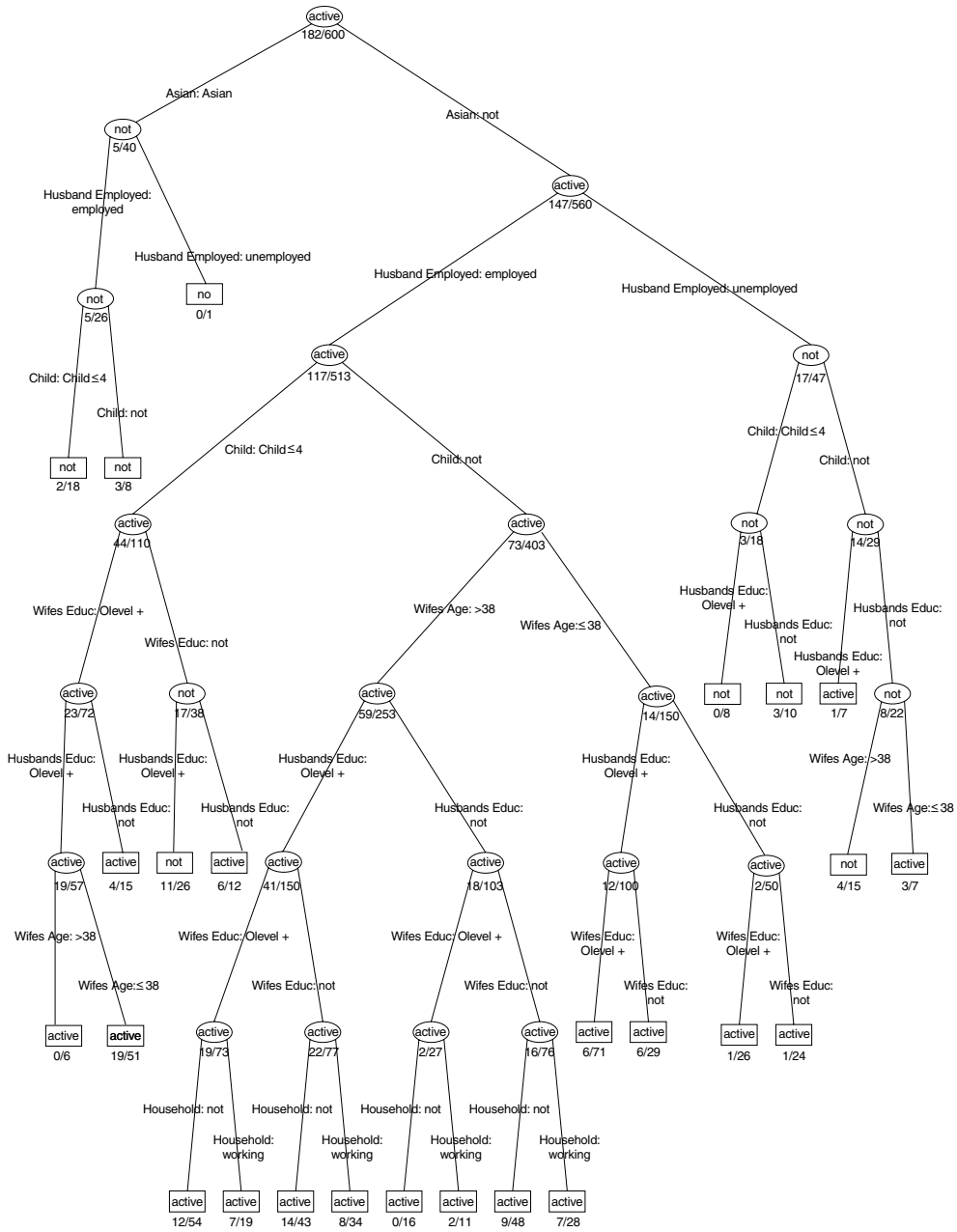


Figure 13.2. An example of a binary classification tree. Goal of the classification was to describe households, in which the wife was economically active. Each branch is labeled according to the current splitting rule. The label in each node indicates the classification taken by majority vote. The numbers below each node give the number of misclassified observations and the total number of observations present in this node

the χ^2 -statistic for contingency tables. The ML family has its origin in computer science, while the CART family is more oriented to statistics using the concept of impurity. Breiman et al. (1984) defined for each node in a tree a measure of *impurity*.

-
- 1 Definition 1** Let c_1, \dots, c_K be the values of the target variable and $P(c_i|n), i = 1, \dots, K$, the (estimated) probability of class c_i in node n (then $\sum_{i=1}^K P(c_i|n) = 1$). The *impurity* of a node n is a nonnegative function $i(n)$ such that
1. $i(n)$ has its only maximum for $P(c_1 | n) = \dots = P(c_K | n) = 1/K$, i.e. the node is as “impure” as possible.
 2. $i(n)$ is minimal if $\exists i \in 1, \dots, K$ such that $P(c_i | n) = 1$, i.e. the node contains only cases, which have the same target value.
-

Some very common splitting criteria based upon impurity measures are:

- Entropy

$$i(n) = - \sum_{j=1}^K P(j|n) \log P(j|n) .$$

- the Gini index of diversity

$$i(n) = \sum_{i \neq j} P(i|n)P(j|n) .$$

Another splitting rule, which is not based on an impurity measure, is the *twoing rule*: A node n is split into a left and a right node n_L and n_R such that

$$\frac{P(n_L)P(n_R)}{4} \left(\sum_{j=1}^K |P(j | n_L) - P(j | n_R)| \right)^2$$

is maximised. For binary targets this rule coincides with the CHAID criterion, which calculates the χ^2 value of a 2×2 table.

As Breiman et al. (1984) point out, the choice of the algorithm used is not as crucial as is generally thought:

within a wide range of splitting criteria the properties of the final tree selected are surprisingly insensitive to the choice of splitting rule (p. 38).

Hand (1997), however, mentioned several problems concerning impurity functions for splitting nodes, for instance

It [the Gini index] has a tendency to produce offspring nodes that are of equal size (p. 69).

Unfortunately, there is no such thing as an optimal splitting criterion. “Optimal” splits very strongly depend on the specific application. When a decision tree is grown, many of the branches will reflect particularities of the training data at hand and will not generalize very well to the test data. This phenomenon is called overfitting, and pruning the tree is a method to address this issue. The prepruning approach tries to implement the pruning process already in the growing phase. For this purpose, an additional stopping criterion is built in. At each node, a split is only performed if the information gain exceeds a certain threshold. Postpruning reduces the complexity of a tree model by removing the branches of some nodes. Postpruning is more effective, especially when the decision to remove a branch is based on a diversity measure that differs from the one used in the growing phase. Prepruning requires less computation but typically does not avoid the problem of overfitting and leads to rather unreliable trees. Postpruning is often a semi-automated process, including manual interactive pruning as well as cross-validation techniques.

Classification trees have been used as a role model for a predictor in bagging (Breiman, 1996). Bagging as well as boosting (Freund and Schapire, 1999) are used to improve the accuracy of a single prediction method by gaining stability and robustness, see also Chap. III.16.

Neural Networks

Artificial neural networks are one of the most prominent data mining techniques. Their fame is two-fold: famous for astonishing good prediction results, unfamous for their black box behavior and lack of reproducibility of achievements. Neural networks are a classical method for predictive modeling. They have been used for classification and prediction to a similar extent. The basic idea of neural networks is the perceptron (see Fig. 13.3), a feedforward neural network with an input layer and an output layer.

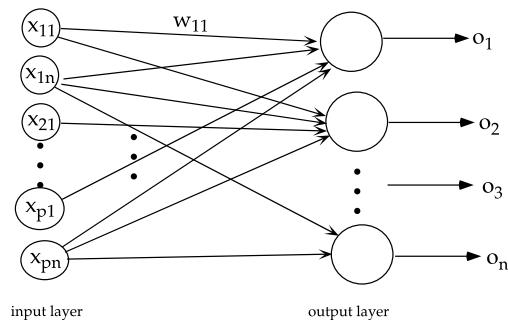


Figure 13.3. Model of a perceptron with n input nodes and n output nodes

The nodes of the input layer serve to introduce the values of the input variables. In a supervised learning situation with p explanatory variables and n cases, we hence get a network with np input nodes and n output nodes. For each output

node $k = 1, \dots, n$ the model output is calculated as a weighted sum of transformed input values

$$o_k = f(a_k) = f\left(\sum_{j=1}^{np} w_{jk}x_j\right).$$

The transformation function f is usually called the activation function. The basic principle of learning a perceptron is the iterative adaptation of the weights w_{jk} in such a way that the error between the observed target values y_i and the model output o_i is as small as possible. The most common delta rule is a gradient method with a tuning parameter η also known as “learning rate.” The choice of η compromises between run time and convergence considerations.

The simple perceptron only allows to solve linearly separable problems. To address more complex problems a multi-layer perceptron, also called feedforward network must be used. A multi-layer perceptron (see Fig. 13.4) introduces additional layers, so-called hidden layers, into the network topology. Each node in the hidden and output layers operates in the same way as an output node in the perceptron. The lack of original target values for the nodes in the hidden layers is remedied by the backpropagation strategy. This means that the present network topology is used in two ways: as a feedforward network to propagate the observed values of the explanatory variables to the output layer and in reverse order to propagate the errors of fitted values back to the input layer.

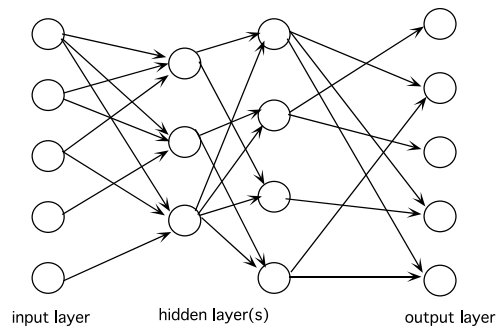


Figure 13.4. Multi-layer perceptron

A huge variety of different models can now be achieved by using different activation functions or different network architectures, i.e. by specifying the links between nodes. Usually, one uses the same activation function for all nodes in the hidden layers. The standard way of network architecture is to fully connect all neurons to all of the units in the preceding layer. However, it is possible to define networks that are partially-connected to only some units in the preceding layer. Typically, the resulting fitted values will be the same (or at least very similar) for different network architectures, summarizing in a nutshell the characteristics of a neural net: often very good in predicting and fitting values but giving very few

insight into the functional relationship between explanatory variables and target variable. Putting it the other way round: neural nets are the method of choice, if you have no idea about the functional relationship between explanatory variables and dependent variable. If you have a strong hypothesis on the functional relationship it is usually preferable to include this knowledge in the modeling process.

Memory Based Reasoning

Memory-Based Reasoning (MBR) tries to mimic human behavior in an automatic way. Memories of specific events are used directly to make decisions, rather than indirectly (as in systems which use experience to infer rules). MBR is a two step procedure: first, identifying similar cases from experience, secondly, applying the information from these cases to new cases. MBR is specifically well suited to non-numerical data. MBR needs a distance measure to assign dissimilarity of two observations and a combination function to combine the results from the neighboring points to achieve an answer. Generating examples is much easier than generating rules which makes MBR so attractive. However, applying rules to new observations is much easier and faster than comparing new cases to a bulk of memorized objects.

Association Rules

Rule induction methods are widely applied tools for mining large data bases. They are often used as a starting point in undirected data mining, i.e. when you do not know what specific patterns to look for. One form of rule induction methods are association rules well-known in market basket analysis. They were proposed by Agrawal et al. (1993) with the intention to provide an automated process, which could find connections among items, that were not known before, especially to answer questions like: “which items are likely to be bought together?”. Many other areas of applications have been named from customer-tailored packages in the telecommunication or insurance business to analyzing web links.

In general, an *association rule* is an implication of the form $X \rightarrow Y$, where X and Y are mutually exclusive item sets. The quality of an association rule $X \rightarrow Y$ is measured by two criteria: confidence and support. A rule holds with *confidence* $c = c(X \rightarrow Y)$, if $c\%$ of transactions in D that contain X also contain Y . The rule $X \rightarrow Y$ has *support* s in the database D , if $s\%$ of transactions in D contain $X \cup Y$.

Most data mining software offers a procedure to generate all association rules with confidence and support that exceed some user-specified minimum thresholds for support (minsup) and confidence (minconf). The procedures are typically based on the a priori algorithm introduced by Agrawal and Srikant (1994).

There are several problems related to this procedure: the setting of the thresholds of minimal support and confidence is crucial; choosing high support and confidence may lead to “uninteresting” results – insofar as the resulting rules are often trivial or well known beforehand by domain experts (Weber, 1998). Lowering the minimal thresholds can lead to a vast increase of the number of results. The standard approach is hence to use low thresholds for generating a large number

of rules and then prune these rules to a manageable number of patterns. Pruning methods are based on objective or subjective interestingness measures. Confidence, support, information gain, Gini-coefficient, entropy, and lift are some of the widely used objective measures (Bayardo and Agrawal, 1999). Subjective measures try to incorporate user and domain knowledge to adapt the resulting rules to the current situation: a pattern which is of interest to one user may be of no interest to another user (Silberschatz and Tuzhilin, 1995). However, it is typically not easy for the user to put his/her expectations and knowledge into automatic procedures of rule pruning. Klemettinen et al. (1994) use rule templates to model the user's knowledge, while Silberschatz and Tuzhilin (1995) introduce belief systems to describe users' expectations.

Another source of problems stems from the inability to estimate the quality of a rule merely from the two keys *confidence* and *support*. Graphical solutions showing a rule in the background of the corresponding contingency table have been proposed by Hofmann et al. (2000). Interactive methods further enhance these displays to powerful tools in the exploration of association rules, see Hofmann and Wilhelm (2001).

13.5.2 Visual Data Mining

Data visualization can contribute to the Data Mining process in many different ways, primarily because the human visual system is excellent in discovering unexpected patterns. Visual data mining does not replace other methods, but it complements analytic approaches. Standard numerical methods to detect outliers and erroneous observations are easier to automate and they may uncover most of the problems in a data set but graphic displays are excellent at identifying and understanding new and unusual problems in the quality of the data. Visualization techniques have met interest in the data base and data mining community since the early 90s, where they have been extensively used to represent results of dynamic data base queries (Shneiderman, 1994; Keim, 1995; Rogowitz et al., 1996; Rogowitz and Treinish, 1996). The parameters of the query are visualized by sliders each representing the range of one query parameter. The user can change the sliders interactively and the query results are shown in a linked graphic. Different methods to represent the distances between the query and the data items have been proposed in the literature: pixel-oriented techniques Keim (1997), different intensities of the highlighting color (Tweddie and Spence, 1998), or the standard linked views approach using a $\{0, 1\}$ -distance (Derthick et al., 1997). More recent discussions of visual approaches to data mining tend to use complex static graphics more suited for presentation than for analysis. (For examples, take a look at websites concerned with Data Mining.) This may be for two reasons. Graphics research in computer science has been concerned with sophisticated, computing-intensive displays (for instance in scientific visualisation) and so it is natural to develop methods of this kind. On the statistical side, commercial software lags substantially behind graphical research. Few packages provide the flexibility and interactivity in graphics that is essential for exploratory work and, of those, even fewer have made provision

for the display and graphical investigation of large data sets. Looking at the scatterplots produced by software for large numbers of data points can reveal more about the software than about the data. The capabilities of graphic displays for initial explorations of a data set are past comparison. Graphic exploration to grasp the main structural features and to get to know the data before beginning formal analysis can be carried out swiftly and informatively. It is not just the graphic displays themselves that are necessary, but the ability to directly work with them: to query points, symbols or axes; to select and link cases across displays; to sort and group data; to rescale and zoom. Interactivity not only means that the user can interact with the data, but also that the results from the changes made by the user can be seen instantaneously. A rapid and responsive interaction facilitates active exploration in a manner that is inconceivable with static displays. Users can start to pose “What if” queries spontaneously as they work through a task. Therefore, interactive displays not only offer the possibility of comparing resulting static views of different aspects of the data, they even encourage to draw conclusions from the way things are changing. Visualisation is also valuable for checking, filtering and comparing results from analytical procedures, and communication of the final outcome to the data base owner and the decision makers is indispensable without charts. At all these stages of the knowledge discovery process, at which contact with domain specialists is important to turn data into knowledge, the advantages of graphical presentation of ideas to enhance communication are considerable.

Research on interactive principles for statistical graphics can be categorized into two classes: firstly, development of innovative tools that help making a single display flexible and dynamic, and secondly, development of tools that operate on the underlying data and therefore have impacts to all displays showing the same data. Common tools of the first class are for example interactive modifiers of the bar width of a histogram, zooming in dot or scatter plots as well as slider-controlled dynamic changes in a graphic. The core of the second class are selection mechanisms and linking. Various selection modes that can even be combined to a sequence help in choosing a specific set of data points to assign interest to them. Linking is the basic concept giving graphical methods the capability of multivariate analysis. Linking builds up a relation between different observations and between different displays. It propagates the changes and selections made in one plot to all other connected plots that deal with the same database.

Visualising Large Numbers of Cases

Data Mining is statistics at scale and speed. Large data sets can be large in two different aspects. First, if the size of the sample investigated is fairly large it will result in a large quantity of observations, the number of data points possibly going towards the billions. Secondly, a large data set can also arise from investigations with a large number of variables. For graphical as well as for analytical procedures both these issues pose problems and require new approaches for solution.

Graphical displays are often used to provide an easy overview of the data from which the global structure can be rapidly deduced while it is still possible at the

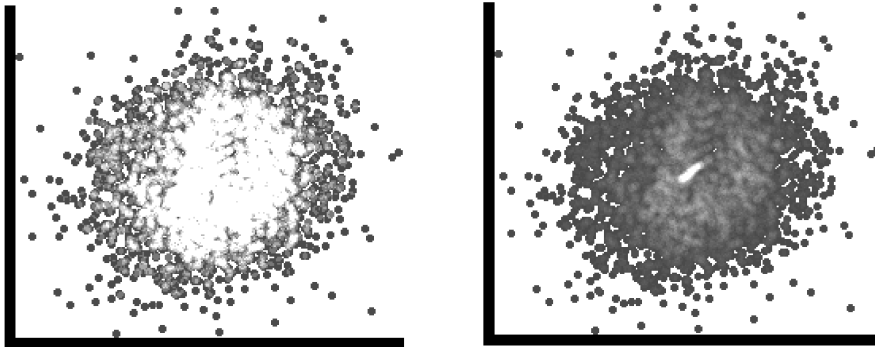
same time to spot individual features. For small data sets one can represent each data point by a single graphical element, as for example in scatter plots in the form of small circles, to ensure that all the information in the data is also represented in the display. However, the use of point based displays reaches its limits as the number of observations increases. A computer screen with about 1280×1024 pixels screen size will possibly offer about one million pixels to be used for displaying points the others needed for frames, scroll bars, legends, and scales. Assuming that we need about five pixels to make a point clearly visible a natural limit for point based displays would lie at about 200,000 observations. This number decreases even more if we take into account that the more structure a data set shows the less spread out are the points over the display's range. Thus, most realistic data sets will only use about a quarter of the available area in a graphic and thus induce that we only can show about 50,000 observations in one point-based plot.

Analytical methods reduce the dimensions and in the extreme case condense the data into one single number, like the mean or a regression coefficient. The graphical analogue is to use one single graphical element to represent the data. A single smooth density curve for example can be used to display the univariate distributional properties of data. Note here, that as in this example, a single graphical element can still convey much more information than a single number. The use of smooth density curves is not only marked by a reduction in terms of graphical elements but also by a transition from using position to encode information to using area instead. Histograms have a long tradition in statistics for being used to display distributional information. They are computationally simpler than smooth density curves and indicate their local constructional properties more clearly. So, for large data sets, area based graphics are to be preferred.

Graphical elements overlap when the values they represent fall too close together on the scale used in the current display. Brightness of the graphical elements can be used to visualize the extent of overplotting. The more graphical elements overlap the brighter they are drawn. One implementation of this is tonal highlighting as provided for dotplots and scatterplots in MANET (Hofmann, 2000). This procedure can be seen as a special kernel density estimation using a uniform kernel. Tonal highlighting and its visual effect is based on two parameters that can be interactively varied: the size of the graphical points used for displaying data and the maximum brightness parameter that is the number of data points that have to overlap to yield the maximum brightness.

For sparse plots brightness should change already with little overplotting, for dense plots a stronger overplotting is needed to cause a change in brightness, see Fig. 13.5.

In contrast to point oriented displays in which dense areas cause problems due to overplotting, it is more likely that area based displays run into difficulties when regions have too few data points. The counts for such regions might be too small to result in an area that is perceptible to the human eye. Especially, histograms for highly unequally dense regions will cause representational problems: according to the scale used either the high density regions are visible or the low density regions.



x-axis is 'Ridge' , y-axis is 'Nub'.

x-axis is 'Ridge' , y-axis is 'Nub'.

Figure 13.5. Tonal highlighting is used to show the amount of overplotting. Increasing the brightness parameter with the overall density of the plot points towards areas with relatively high overplotting. The *left plot* shows the default setting. The *plot on the right* with increased brightness parameter clearly shows a pencil shape region in the center with a very high density

Linking leads to a technical difficulty associated with large data sets and particularly relevant to data quality. Even with a high-resolution screen it may be necessary for each pixel to represent many points. If the number of points to be drawn or highlighted is too small to be represented, then those points will be invisible. This can lead to false judgments, for instance in masking unusual cases, and needs to be avoided. One solution, which works well in practice, has been implemented in the software MANET. Red-marking is used to indicate that some information may be hidden. For histograms and barcharts a red line is drawn under any bar where information may be hidden. It could be that the bar itself is too small to be drawn (which can happen with extreme values in histogram displays); it could be that too few are highlighted in a bar for any highlighting to show; it could be that too few are not highlighted in a bar so that the whole bar is, misleadingly, completely highlighted. A related problem arises in the display of spatial data: very small areas (often the most populous!) may not be shown and special red-marking techniques are needed to take account of this.

Direct manipulation graphics can alleviate this problem further when there is the possibility of easily changing the scale of a display. Here it is by no means sufficient to have a parameter tool box in which we can insert our desired scaling. We must be able to grab the vertical axis in a histogram to shrink or expand a scale by simply dragging it as well as breaking a scale and using different scalings at different parts of the range. Such a feature can be thought of as a particular variant of logical zooming. While standard zooming enlarges the displayed graphical elements, logical zooming works on the underlying model and changes it to display more details. Logical zooming is quite natural when working with maps, starting with a country map, we zoom into a regional map, a city map and finally a street map that shows us the neighborhood in every detail.

Logical zooming for the vertical axis aims at balancing out the needs for a good overview and the need for focussing on individual features. Area based displays like histograms aggregate the data and lose the ability to show anomalous behavior of single points. For large data sets this is not an unwelcome feature since huge data sets might contain too many features to investigate. So the researcher will only focus on phenomena that occur frequently enough.

Logical zooming is even more important when used on the x -axis to change the level of aggregation. Again, there is not only the need to zoom in a homogeneous way such that all bins of a histogram are treated in the same manner and use the same scaling for all of them. Rather often, the analyst might be interested in particular regions, for example in the center of a distribution to see whether there are any gaps that just might be smoothed away by the graphical representation. Thus, a very common interest is zooming into one region while still keeping the general overview.

Logical zooming is valuable for all statistical displays that aggregate the data: for boxplots, for example, logical zooming should result in displaying the dotplot for the selected section or a boxplot based on the selected points only. Logical zooming in mosaic plots would help investigating how additional variables could split up a large cell. Hot selection as provided in DataDesk (Velleman, 2000) can be seen as an elementary form of logical zooming.

Visualising Large Numbers of Variables

Common orthogonal coordinate systems have at most three orthogonal axes for visualization at hand. To display more variables and typically most data sets will comprise a much larger number of variables projection techniques are widely used to reduce the dimensionality to a manageable size. Other approaches are using matrix layouts for scatterplots and providing brushing techniques to visually connect points that belong together, but this only works for a low number of dimensions. Another approach is to use parallel coordinate systems that can deal with a larger number of variables simultaneously. The parallel coordinate display (Wegman, 1990; Inselberg, 1985) sacrifices orthogonal axes by drawing the axes parallel to each other resulting in a planar diagram where each d -dimensional point (x_1, \dots, x_d) is uniquely represented by a broken line. Current screens limit the numbers of variables that can be simultaneously shown in parallel coordinate plots to about 30 – but, of course, scrolling windows offer in principle unlimited capabilities.

The individual parallel coordinate axes represent one-dimensional projections of the data. Usually, different variables will be based on different units of measurement. Using the original scale might make inefficient use of the screen space. Using standardized variables will ameliorate that. In certain instances the original scale will, however, display valuable information. An easy interactive change of this scaling is thus a particularly welcome feature. Pairwise relationships for adjacent variables are much more easily seen than for nonadjacent variables. Since a complete parallel coordinate investigation would require running through all possible

permutations, or interactive facilities for manually or automatically changing the order of the variables are needed.

Due to the point – line duality between a parallel coordinate system and a cartesian coordinate system the correlation of two adjacent variables is depicted by the mutual position of the line segments: parallel line segments indicate a strong positive correlation, a line crossing in a single point means strong negative correlation. Since it is much easier to recognize an intersection point than almost parallel lines, negative correlation is simpler to detect. It is helpful to have an interactive tool to invert the range of a single variable to turn positive correlation into a negative one. CASSATT is a JAVA application that offers a wide variety of interactive features for parallel coordinate plots, see Winkler (2000). Linking and highlighting interactive statistical graphics increases the dimensionality of data that can be explored. For highly multivariate data (i.e., more than ten to twenty variables) insight into the data by linking low-dimensional plots can be limited. Thus the need for high-dimensional plots arises. These plots – for example, rotating plots (grand tour, projection pursuit, see Chap. II.10), parallel coordinate plots, or mosaic plots – can incorporate up to ten or more variables in a single plot. Linked highlighting and alterations inside these plots (e.g., zooming, reordering, or sorting) offer high-dimensional insights into data sets. Multiple selections via selection sequences offer a convenient way of interacting with high-dimensional subsets of the data using low-dimensional plots.

Visualizing Association Rules

Visualizing association rules aims at solving some major problems that come with association rules. First of all the rules found by automatic procedures must be filtered. Depending on what minimum confidence and what support is specified a vast amount of rules may be generated. Among those, however, not only “interesting and new” results – according to the principle of KDD – are found. In supermarket data for instance most purchases will also contain plastic or paper bags. The naive filtering approach that searches for the rules with highest confidence and/or highest support fails because it yields rules that are typically already known before and are thus not of interest. So, in many cases rules that just do not pass the thresholds can be economically better exploited and are therefore higher rated. Association rules tend to prefer frequent events as their response (right hand side of a rule). If $P(Y)$ is large, then it is very likely for any small event X , that $P(Y|X)$ is higher than the minimal confidence threshold. The meaning of rules found in this way, on the other hand, is more than questionable. When dealing with different association rules which refer to the same response, it is of interest to examine, whether the populations described by the explanatory variables (left hand side of a rule) differ from each other. It well could be one and the same population, and different rules providing only different descriptions for it. – Another goal therefore is, to examine the impact of rules on one another and to find intersections amongst them. Quite often, slight variations of items are listed as different association rules: either the same items alternatively take right and left hand sides of rules or sequences occur.

Besides finding new and interesting results, data mining tools are to find *explainable* results. The interpretation of results therefore should be of great interest – if one can not explain a rule at first, we could use methods of cross-classification in the hope of finding clues within the data, which allow us to decide, whether a result has to be considered as a peculiarity of this particular data set or can be generalized.

Since support and confidence are equally important for the conclusions drawn from association rules any approach should visualize support and confidence of competing rules within one display. Figure 13.6 shows a standard visualization of association rules as a matrix of all left and right hand sides of rules. Left hand sides are the rows, right hand sides the columns of the matrix. Each rule, which fulfills $\text{minsup } p$ and minconf is drawn as a square. The size of which is given by the actual support of the rule. This approach is rather unsatisfactory since it reaches the space limits already for a small number of association rules. It also uses two different visual attributes to encode support and confidence of a rule: color for the confidence and size for the support. The ordering of colors is difficult and in no way unambiguous and depends heavily on the used scale of the color scheme. The encoding of the support by the size of the squares is visually problematic, since length of the square is used instead of area. For instance the rules

turkey & hering & corned beef → *olives*

ham & corned beef & apples → *olives*

have support 11.19% and 3.1%, respectively. The factor is approximately 4, whereas the areas differ with factor 16 which yields a lie factor as defined by Tufte (1983) of 400%.

Mosaic plots as introduced by Hartigan and Kleiner (1981) are a graphical analogue to multivariate contingency tables. They show the frequencies in a contingency table as a collection of rectangles whose areas represent the cell frequencies. Large areas therefore represent large cells. The shape of a tile is calculated during the (strictly hierarchical) construction. In classical mosaic plots alternately width and height of each bin is split according to the categories of the next variable included, in such a way, that the area of each bin is proportional to the number of cases falling into this specific intersection of variables. Thus, viewed statically the mosaic plot gives a graphical estimation of the joint distribution of the variables contained. Interactive features are necessary to turn the mosaic plot into a powerful exploration tool. In Hofmann (1999) these interactive features have been explained with an example, the most essentials are linked highlighting, rotating, navigating through dimensions, grouping categories, and variation of display.

Linked highlighting provides the transition to conditional distributions and is essential for visualizing association rules. The basic approach is as follows: Combine all variables involved in the left-hand-side X of a rule as *explanatory* variables and draw them within one mosaicplot. Visualize the *response* Y , the right-hand-side of the rule, in a bar chart and then by highlighting a category in

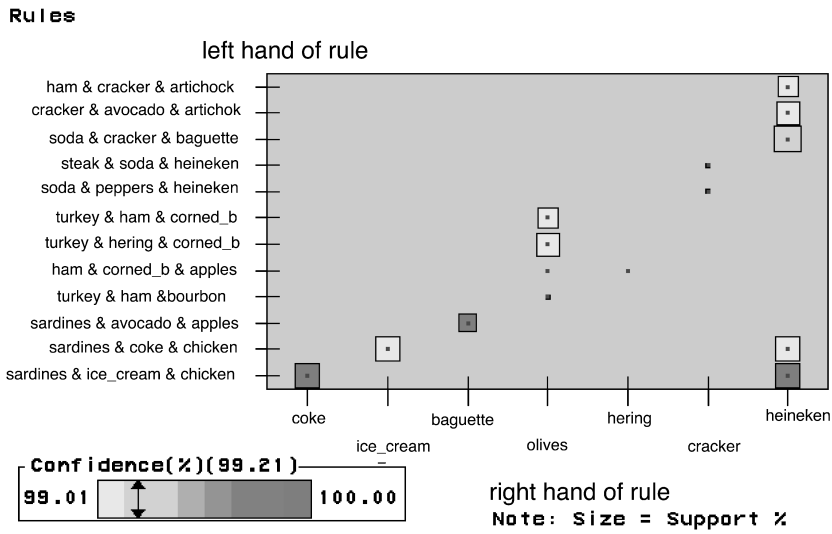


Figure 13.6. SAS Enterprise Miner: Visualisation of all 15 association rules with a minimum confidence of 99%

the bar chart every tile in the mosaic plot corresponds to an association rule. The amount of highlighting in a bin in the mosaic plot gives a visual measure for the support of a rule, the highlighting heights relative to the bin's height give the rule's confidence.

Figure 13.7 shows an overview of all possible association rules involving three binary variables X_1, X_2 and Y . The bin in the bottom right corner ($x_1 \cap x_2 \cap x_3$) gives an example for a rule that passes most common thresholds. It has very high confidence (the highlighting almost fills this bin entirely), and also the support is relatively high (the bin itself, and therefore the amount of highlighting, is

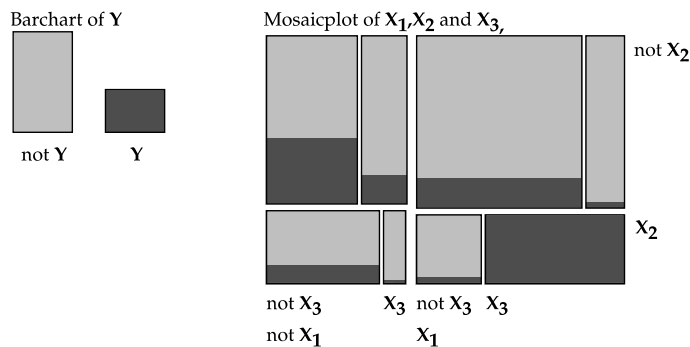


Figure 13.7. Mosaic plot of all possible association rules of X_1, X_2, X_3 and Y . The amount of highlighting in a bin represents the support of a rule, while its confidence is measured by the proportion of highlighting within a tile

large). The leftmost bin in the upper row, ($\text{not } x_1 \cap x_2 \cap \text{not } x_3$), represents a rule with a comparable support (this bin contains approximately the same amount of highlighting as the bin in the bottom right corner), yet the confidence of the corresponding rule is rather low (highlighting fills this bin to only one third, approximately). All the other possible rules have even lower confidence and their supports are also too small to be of interest.

Rotated Mosaic Plots. To make comparisons of proportions of highlighting heights more easily, it is much better to align all bins. This yields mosaics of the following form: Starting from the basic rectangle, the width is divided according to the first variable's categories and their numbers of entries. Each of these bins is divided again and again horizontally in the same way according to the other variables. In MANET standard mosaic plots can be interactively rotated in this form by a simple mouse click and therefore we call these plots rotated mosaic plots. Highlighting splits the bins still vertically. Thus highlighting heights in a p dimensional mosaic plot of variables X_1, \dots, X_p show the conditional probabilities $P(h|X_1, \dots, X_p)$.

In addition, to determine more easily the exact combination that a bin represents labels can be added underneath the mosaics (see Fig. 13.8). Each row of the labels corresponds to one variable, the white rectangles stand for "0"s, the blacks for "1"s. The first bin in Fig. 13.8 therefore represents the combination of "no item bought" (all "0"s), the second bin contains all transactions, where only *corned beef* has been bought, and so on. This form of diagrams is also known as doubledecker plots introduced in Hofmann et al. (2000).

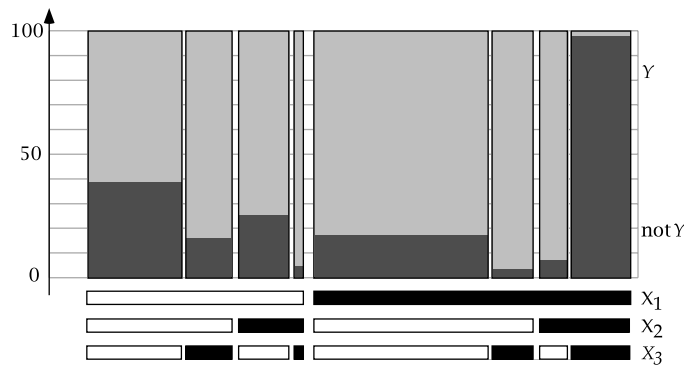


Figure 13.8. Doubledecker plot of three variables. The rightmost bin corresponds to the rule *heineken & coke & chicken* \rightarrow *sardines*

References

- Agrawal, R., Imielinski, T., and Swami, A. (1993). Mining associations between sets of items in massive databases. In *Proceedings of the ACM-SIGMOD 1993 International Conference on Management of Data*, pages 207–216, Washington D.C.
- Agrawal, R. and Srikant, R. (1994). Fast algorithms for mining association rules. Technical Report RJ9839, IBM. IBM Research Report RJ9839.
- Bayardo, R. J. and Agrawal, R. (1999). Mining the most interesting rules. In *Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 145–154.
- Brachman, R. and Anand, T. (1996). The process of knowledge discovery in databases. In *Advances in Knowledge Discovery and Data Mining*, pages 37–57. AAAI Press/The MIT Press.
- Breiman, L. (1996). Bagging predictors. *Machine Learning*, 24(2):123–140.
- Breiman, L., Friedman, J. H., Olshen, R., and Stone, C. (1984). *Classification and Regression Trees*. Wadsworth, Belmont, CA.
- Derthick, M., Kolojejchick, J., and Roth, S. F. (1997). An interactive visualization environment for data exploration. Technical report, Carnegie Mellon University, Pittsburgh, PA.
- Fayyad, U., Piatetsky-Shapiro, G., Smyth, P., and Uthurusamy, R., editors (1996). *Advances in Knowledge Discovery and Data Mining*. AAAI/MIT Press.
- Freund, Y. and Schapire, R. (1999). A short introduction to boosting.
- Friedman, J.H. (1998). Data mining and statistics: What's the connection? In *Computing Science and Statistics: Proceedings of the 29th Symposium on the Interface*. Interface Foundation of North America.
- Hand, D. (1997). *Construction and Assessment of Classification Rules*. John Wiley & Sons, Chichester.
- Hand, D.J. (1998). Data mining: statistics and more? *American Statistician*, pages 112–118.
- Hand, D.J., Mannila, H., and Smyth, P. (2001). *Principles of Data Mining*. MIT Press.
- Härdle, W. (1991). *Smoothing Techniques with implementation in S*. Springer, New York.
- Hartigan, J. and Kleiner, B. (1981). Mosaics for contingency tables. In *Computer Science and Statistics: Proceedings of the 13th Symposium on the Interface*, pages 268–273, New York. Springer Verlag.
- Hofmann, H. (1999). Simpson on board the Titanic? *Statistical Computing and Graphics Newsletter*, 9.
- Hofmann, H. (2000). MANET. <http://stats.math.uni-augsburg.de/manet>.
- Hofmann, H., Siebes, A., and Wilhelm, A. (2000). Visualizing association rules with interactive mosaic plots. In *Proceedings of the 6th International Conference on Knowledge Discovery and Data Mining*, pages 227–235, Boston MA.

- Hofmann, H. and Wilhelm, A. (2001). Visual comparison of association rules. *Computational Statistics*, 16:399–415.
- Inselberg, A. (1985). The plane with parallel coordinates. *The Visual Computer*, 1:69–91.
- Keim, D.A. (1995). Enhancing the visual clustering of query-dependent data visualization techniques using screen-filling curves. In *Proc. Int. Workshop on Database Issues in Data Visualization*.
- Keim, D.A. (1997). Visual techniques for exploring databases. *Tutorial Notes: Third International Conference on Knowledge Discovery and Data Mining*, pages 1–121.
- Klemettinen, M., Mannila, H., Ronkainen, P., Toivonen, H., and Verkamo, A. (1994). Finding interesting rules from large sets of discovered association rules. In *Proceedings of the Third International Conference on Information and Knowledge Management CIKM-94*, pages 401–407.
- Kloesgen, W. and Zytkow, J.M. (1996). Knowledge discovery in database terminology. In *Advances in Knowledge Discovery and Data Mining*, pages 573–592.
- Rogowitz, B.E., Rabenhorst, D.A., Gerth, J.A., and Kalin, E.B. (1996). Visual cues for data mining. Technical report, IBM Research Division, Yorktown Heights, NY.
- Rogowitz, B.E. and Treinish, L.A. (1996). How not to lie with visualization. *Computers in Physics*, 10:268–273.
- Scott, D.W. (1992). *Multivariate Density Estimation*. Wiley, New York.
- Shneiderman, B. (1994). Dynamic queries for visual information seeking. *IEEE Software*, 11(6):70–77.
- Silberschatz, A. and Tuzhilin, A. (1995). On subjective measures of interestingness in knowledge discovery. In *Proceedings of the First International Conference on Knowledge Discovery and Data Mining*, pages 275–281.
- Tufte, E.R. (1983). *The Visual Display of Quantitative Information*. Graphics Press, Cheshire, CT.
- Tukey, J.W. (1969). Analyzing data: Sanctification or detective work? *American Psychologist*, 24:83–91.
- Tweedie, L. and Spence, R. (1998). The prosecution matrix: A tool to support the interactive exploration of statistical models and data. *Computational Statistics*, 13:65–76.
- Unwin, A., Hofmann, H., and Wilhelm, A. (2002). Direct manipulation graphics for data mining. *International Journal of Image and Graphics*, 2:49–65.
- Velleman, P. (2000). DataDesk. <http://www.datadesk.com>.
- Weber, I. (1998). On pruning strategies for discovery of generalized and quantitative association rules. In *Proceedings of Knowledge Discovery and Data Mining Workshop*, Singapore.
- Wegman, E.J. (1990). Hyperdimensional data analysis using parallel coordinates. *Journal of the American Statistical Association*, 85:664–675.
- Winkler, S. (2000). Parallele Koordinaten: Entwicklung einer interaktiven Software – CASSATT. Technical report, University of Augsburg. in German.

Recursive Partitioning and Tree-based Methods

III.14

Heping Zhang

14.1	<i>Introduction</i>	814
14.2	<i>Basic Classification Trees</i>	817
	Tree Growing and Recursive Partitioning	817
	Tree Pruning and Cost Complexity	818
14.3	<i>Computational Issues</i>	820
	Splits Based on an Ordinal Predictor	820
	Splits Based on a Nominal Predictor	823
	Missing Values	824
14.4	<i>Interpretation</i>	825
14.5	<i>Survival Trees</i>	828
	Maximizing Difference Between Nodes	829
	Use of Likelihood Functions	830
	A Straightforward Extension	831
	Other Developments	831
14.6	<i>Tree-based Methods for Multiple Correlated Outcomes</i>	832
14.7	<i>Remarks</i>	833

Introduction

Tree-based methods have become one of the most flexible, intuitive, and powerful data analytic tools for exploring complex data structures. The applications of these methods are far reaching. They include financial firms (credit cards: Altman, 2002; Frydman et al., 2002, and investments: Pace, 1995; Brennan et al., 2001), manufacturing and marketing companies (Levin et al., 1995), and pharmaceutical companies.

The best documented, and arguably most popular uses of tree-based methods are in biomedical research for which classification is a central issue. For example, a clinician or health scientist may be very interested in the following question (Goldman et al., 1996; Goldman et al., 1982; Zhang et al., 2001): Is this patient with chest pain suffering a heart attack, or does he simply have a strained muscle? To answer this question, information on this patient must be collected, and a good diagnostic test utilizing such information must be in place. Tree-based methods provide one solution for constructing the diagnostic test.

Classification problems also frequently arise from engineering research. Bahl et al. (1989) introduced a tree-based language model for natural language speech recognition. Desilva and Hull (1994) used the idea of decision trees to detect proper nouns in document images. Geman and Jedynak (1996) used a related idea to form an active testing model for tracking roads in satellite images. In addition, decision trees have been used in scientific and social studies including astronomy (Owens et al., 1996), chemistry (Chen et al., 1998) and politics (<http://www.dtreg.com/housevotes.htm>). We will revisit some of these applications later in detail.

Most commercial applications of tree-based methods have not been well-documented through peer reviewed publications. In 1999 the author helped the CLARITAS, a marketing company, apply a tree-based method as described in Sect. 14.6 (Zhang, 1998) for marketing segmentation analysis. Tree-based methods have also been frequently used in the drug development process. The author has personally provided consultations to Aventis, Inc. for drug approvals.

The purpose of this article is to provide an overview for the construction of the decision trees, and, particularly, the recursive partitioning technique, which is the thrust of this methodology. In their early applications, tree-based methods were developed primarily to facilitate the automation of classifications as an expert system (Breiman et al., 1984; Friedman, 1977; Wasson et al., 1985), although Morgan and Sonquist (1963) were motivated by the need to analyze survey data to identify interactions, particularly in the presence of non-numerical predictors. More recently, classification trees have not only been used for automated disease diagnosis, but also for selecting important variables that are associated with a disease or any response of interest (Zhang and Bracken, 1995; Zhang and Bracken, 1996; Zhang and Singer, 1999; Zhang et al., 2003; Zhang et al., 2001).

There are different approaches to classification. First, it can be done intuitively. For example, a physician or a group of physicians may use their experience in

caring for patients with chest pain to form a subjective opinion or an empirical decision as to whether a new patient with chest pain is likely to suffer a heart attack, and consequently, decide what treatment is most appropriate. Secondly, methods in both statistical and machine learning literature have been developed, such as Fisher linear discriminant analysis (Fisher, 1936) and support vector machine (Cristianini and Shawe-Taylor, 2000). These methods have the parametric flavor in the sense that the classification rule has an explicit form with only a few parameters to be determined from a given sample that is usually referred to as learning sample.

Classification trees belong to the third type of methods for which we allow a very general structure, e.g., the binary tree as displayed in Fig. 14.1, but the number of “parameters” also needs to be determined from the data, and this number varies. For this reason, classification trees are regarded as nonparametric methods. They are adaptive to the data and are flexible, although the large number of quantities (or parameters) to be estimated from the data makes the classification rule more vulnerable to noise in the data.

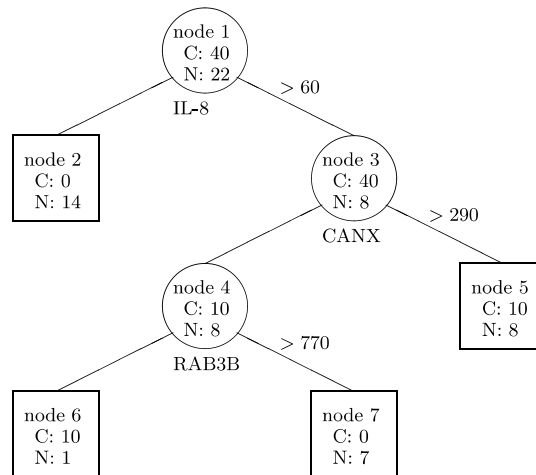


Figure 14.1. Classification Tree for Colon Cancer Diagnosis Based on Gene Expression Data. Inside each node are the number of tumor (C) and normal (N) tissues. See Zhang et al. (2001) for more details

To be more precise about the statistical problem, let us define the data structure and introduce some notation. Suppose that we have observed p covariates, denoted by a p -vector \mathbf{x} , and a response y for n individuals. For the i th individual, the measurements are

$$\mathbf{x}_i = (x_{i1}, \dots, x_{ip})' \quad \text{and} \quad y_i, \quad i = 1, \dots, n.$$

The objective is to model the probability distribution of $P(y|\mathbf{x})$ or a functional of this conditional distribution.

To appreciate how these variables are characterized in real applications, let us examine some of the published applications.

- 1 **Example 1** Levin et al. (1995) described a probability-driven, customer-oriented decision support system for the marketing decisions of the Franklin Mint, a leading Philadelphia-based worldwide direct response marketer of quality collectibles and luxury home decor products. The purpose of the system is to target the “right” audience for each promotion from among a very large marketing database, based on the customers’ attributes and characteristics. In this case, the customers’ attributes and characteristics constitute the x variables. Whether the targeted client is desirable or not forms the basis for the response y .
-

- 2 **Example 2** To screen large chemical databases in corporate collections and chemical libraries, Chen et al. (1998) used recursive partitioning to develop three-dimensional pharmacophores that can guide database screening, chemical library design, and lead optimization. Their idea was to encode the three-dimensional features of chemical compounds into bit strings, and those features are the x variables. Then, those features are selected in relation to the biological activities (i.e., y) of the compounds. Here, each compound contributes an observation. Using this idea, the authors successfully retrieved three-dimensional structure-activity relationships from a large heterogeneous dataset of 1644 monoamine oxidase inhibitors. We will revisit this example in detail in Sect. 14.4.
-

Like any multivariate regression model and as we can see from the above examples, the covariates or predictors in x may contain variables that can be categorical (nominal or ordinal) or continuous. For example, ethnicity is usually treated as categorical data and age as continuous. Some of the covariates may have missing values, and we will discuss how missing values are handled in the tree framework. In a nutshell, unlike what is usually done in a simple linear regression, observations with missing information are not omitted from classification trees.

Not only can we have mixed types of predictors, but also the response variable can be discrete (binary or multiclass), continuous, and sometimes censored. The characteristics of the response, y , determines the method for estimating $P(y | x)$. We will review a variety of tree-based methods that are adaptable to the distribution of y . In Sect. 14.2, we will introduce the basic idea of classification trees using a dichotomous response. Section 14.2 is followed by some in-depth discussion of computational challenges and implementations in Sect. 14.3 and by examples in Sect. 14.4 to illustrate how we can interpret results from tree-based analyses. One of the most popular uses of tree-based methods is in the analysis of censored data in which y is the time to an event and is subject to censoring. As described in Sect. 14.5, such trees are referred to as survival trees (Bacchetti and Segal, 1995; Carmelli et al., 1991; Carmelli et al., 1997; Gordon and Olshen, 1985; Zhang, 1995). In Sect. 14.6, we will present an extension of the tree methodology to the classification

of a response consisting of multiple components such as an array of respiratory symptoms (Zhang, 1998). Finally, we will conclude in Sect. 14.7 with some remarks on relatively recent developments such as forests and Bayesian trees. To illustrate the methods and their applications, some examples will be presented along with the methods.

Basic Classification Trees

14.2

We have highlighted some applications of decision trees. Here, we will explain how they are constructed. There has been a surge of interest lately in using decision trees to identify genes underlying complex diseases. For this reason, we will begin the explanation of the basic idea with a genomic example, and then will also discuss other examples.

Zhang et al. (2001) analyzed a data set from the expression profiles of 2000 genes in 22 normal and 40 colon cancer tissues (Alon et al., 1999). In this data set, the response y equals 0 or 1 according to whether the tissue is normal or with cancer. Each element of x is the expression profile for one of the 2000 genes. The objective is to identify genes and to use them to construct a tree so that we can classify the tumor type according to the selected gene expression profiles. Figure 14.1 is a classification tree constructed from this data set. In what follows, we will explain how such a tree is constructed and how it can be interpreted.

Tree Growing and Recursive Partitioning

14.2.1

Tree construction usually comprises two steps: growing and pruning. The growing step begins with the root node, which is the entire learning sample. In the present example, the root node contains the 62 tissues and it is labeled as node 1 on the top of Fig. 14.1. The most fundamental step in tree growing is to partition the root node into two subgroups, referred to as daughter nodes, such that one daughter node contains mostly cancer tissue and the other daughter node mostly normal tissue. Such a partition is chosen from all possible binary splits based on the 2000 gene expression profiles via questions like “Is the expression level of gene 1 greater than 200?” A tissue is assigned to the right or left daughter according to whether the answer is yes or no. When all of the 62 tissues are assigned to either the left or right daughter nodes, the distribution in terms of the number of cancer tissues is assessed for both the left and right nodes using typically a node impurity. One of such criteria is the entropy function

$$i_t = -p_t \log(p_t) - (1 - p_t) \log(1 - p_t),$$

where p_t is the proportion of cancer tissue in a specified node t . This function is at its lowest level when $p_t = 0$ or 1. In other words, there is the least impurity when the node is perfect. On the other hand, it reaches the maximum when $p_t = 1/2$, that is, the node is equally mixed with the cancer and normal tissues.

Let L and R denote the left and right nodes, respectively. The quality of the split s , resulting from the question “Is the expression level of gene 1 greater than 200?” is measured by weighing i_L and i_R as follows:

$$g_s = 1 - Pr(L)i_L - Pr(R)i_R, \quad (14.1)$$

where $Pr(L)$ and $Pr(R)$ are probabilities of tissues falling into the left and right nodes, respectively. The split with the lowest g_s is ultimately chosen to split the root node. This very same procedure can be applied to split the two daughter nodes, leading to the so-called recursive partitioning process. This process dies out as the sizes of the offspring nodes become smaller and smaller and the distribution of the tissue type becomes more and more homogeneous. The splitting stops when the node contains only one type of tissues.

The objective of the tree growing step is to produce a tree by executing the recursive partitioning process as far as possible. A natural concern is that such a saturated tree is generally too big and prone to noise. This calls for the second step to prune the saturated tree in order to obtain a reasonably sized tree that is still discriminative of the response whereas parsimonious for interpretation and robust with respect to the noise.

14.2.2 Tree Pruning and Cost Complexity

For the purpose of tree pruning, Breiman et al. (1984) introduced misclassification cost to penalize the errors of classification such as classifying a cancer tissue as a normal one, and vice versa. The unit of misclassification cost is chosen to reflect the seriousness of the errors because the consequence of classifying a cancer tissue as a normal one is usually more severe than classifying a normal tissue as a cancer one. A common practice is to assign a unit cost for classifying a normal tissue as a cancer one and a cost, c , for classifying a cancer tissue as a normal one. Once c is chosen, the class membership for any node can be determined to minimize the misclassification cost. For example, the root node of Fig. 14.1 is classified as a cancer node for any c chosen to be greater than 22/40. While c is usually chosen to be greater than 1, for the purpose of illustration here, if it is chosen to be 0.5, the root node is classified as a normal node because it gives rise to a lower misclassification cost.

When the class memberships and misclassification costs are determined for all nodes, the misclassification cost for a tree can be computed easily by summing all costs in the terminal nodes. A node is terminal when it is not further divided, and other nodes are referred to as internal nodes. Precisely, the quality of a tree, denoted by T , is reflected by the quality of its terminal nodes as follows:

$$R(T) = \sum_{t \in \tilde{T}} Pr(t)R(t), \quad (14.2)$$

where \tilde{T} is the set of terminal nodes of tree T and $R(t)$ the within-node misclassification cost of node t .

The ultimate objective of tree pruning is to select a subtree of the saturated tree so that the misclassification cost of the selected subtree is the lowest on an independent, identically distributed sample, called a test sample. In practice, we rarely have a test sample. Breiman et al. (1984) proposed to use cross validation based on cost-complexity. They defined the number of the terminal nodes of T , denoted by $|\tilde{T}|$, as the complexity of T . A penalizing cost, the so-called complexity parameter, is assigned to one unit increase in complexity, i.e., one extra terminal node. The sum of all costs becomes the penalty for the tree complexity, and the cost-complexity of a tree is:

$$R_\alpha(T) = R(T) + \alpha|\tilde{T}|, \quad (14.3)$$

where $\alpha(> 0)$ is the complexity parameter.

A useful and interesting result from Breiman et al. (1984) is that, for a given complexity parameter, there is a unique smallest subtree of the saturated tree that minimizes the cost-complexity measure (14.3). Furthermore, if $\alpha_1 > \alpha_2$ the optimally pruned subtree corresponding to α_1 is a subtree of the one corresponding to α_2 . Therefore, increasing the complexity parameter produces a finite sequence of nested optimally pruned subtrees, which makes the selection of the desirably-sized subtree feasible.

Although the introduction of misclassification cost and cost complexity provides a solution to tree pruning, it is usually a subjective and difficult decision to choose the misclassification costs for different errors. Moreover, the final tree can be heavily dependent on such a subjective choice. From a methodological point of view, generalizing the concept of misclassification cost is difficult when we have to deal with more complicated responses, which we will discuss in detail later. For these reasons, we prefer a simpler way for pruning as described by Segal (1988) and Zhang and Singer (1999).

Let us now return to the example. In Fig. 14.1, the 62 tissues are divided into four terminal nodes 2, 5, 6, and 7. Two of them (Nodes 2 and 7) contain 21 normal tissues and no cancer tissue. The other two nodes (Node 5 and 6) contain 40 cancer tissues and 1 normal tissue. Because this tree is relatively small and has nearly perfect classification, pruning is almost unnecessary. Interestingly, this is not accidental for analyses of many microarray data for which there are many genes and relatively few samples.

The construction of Fig. 14.1 follows the growing procedure as described above. First, node 1 is split into nodes 2 and 3 after examining all allowable splits from the 2000 gene expression profiles, and the expression level of gene IL-8 and its threshold at 60 are chosen because they result in the lowest weighted impurity of nodes 2 and 3. A tissue is sent to the left (node 2) or right (node 3) daughter node according to whether or not the IL-8 level is below 60. Because node 2 is pure, no further split is necessary and it becomes a terminal node. Node 3 is split into nodes 4 and 5 through recursive partitioning and according to whether or not the expression of gene CANX is greater than 290, while the partition is restricted to the 40 tissues in node 3 only. Furthermore, node 4 is subsequently partitioned

into nodes 6 and 7 according to whether or not the expression of gene RAB3B exceeds 770.

There are also many interesting applications of simple classification trees. For example, Goldman et al. (1982) used classification trees to predict heart attack based on information from 482 patients. After a tree is constructed, the prediction is made from a series of questions such as “Is the pain in the neck only?” and/or “Is the pain in the neck and shoulder?” An appealing feature of tree-based classification is that the classification rule is based on the answers to simple and intuitive questions as posed here.

Although we present classification trees for a binary response, the method is similar for a mult-level response. The impurity function can be defined as

$$i_t = - \sum_{j=1}^J Pr(y = j) \log\{Pr(y = j)\} ,$$

for a J -level y . Everything else in the tree growing step as described above is applicable. For tree pruning, the only change to be made is to define the misclassification cost $c(j|k)$ from level k to level j , $j, k = 1, \dots, J$.

14.3 Computational Issues

In Sects. 14.2.1 and 14.2.2, we have explained the basic steps and concepts for tree construction. For most users of decision trees, the implementation aspect does not really affect the application. For methodological and software developments, however, it is imperative to understand the computational issues. The most critical issue is to find the optimal split efficiently for any given node. The overall strategy is to identify the optimal split from each of the predictors and then choose the overall best one. Choosing the overall best one is straightforward, but identifying the optimal split from a predictor takes some efforts. The algorithm must take into account the nature of the predictor. Although we will use a dichotomous response to explain the ideas, the algorithm is also applicable for the other types of responses.

14.3.1 Splits Based on an Ordinal Predictor

Let us first consider a predictor with an ordinal scale such as gene expression in Fig. 14.1 or the ratio of cash flow to total debt in Fig. 14.2. Under the tree framework, as long as a predictor is ordinal, we will soon see that it does not matter whether the predictor is on a continuous or discrete scale.

Table 14.1 displays the expression levels of gene IL-8 in 22 normal and 40 colon cancer tissues. Our objective for the time being is to split these 62 tissues into two subsamples according to whether the expression level of gene IL-8 is greater than a given threshold. In theory, this threshold can be anything, but practically, there

Table 14.1. Expression level of gene IL-8 in 22 normal and 40 colon cancer tissues used in Fig. 14.1

Expression level	Colon cancer	Expression level	Colon cancer	Expression level	Colon cancer	Expression level	Colon cancer
23.74	N	35.95875	N	33.9725	N	45.1	N
56.91875	N	28.7675	N	28.00875	N	39.7575	N
11.37625	N	31.6975	N	30.57875	N	171.4525	N
36.8675	N	40.33875	N	76.9875	N	97.92	N
55.2	N	238.58625	N	645.99375	N	117.6025	N
113.91375	N	567.13125	N	1528.4062	Y	306.30875	Y
76.125	Y	169.1375	Y	213.6275	Y	326.42625	Y
370.04	Y	114.92375	Y	311.4375	Y	186.2775	Y
131.65875	Y	412.135	Y	284.14625	Y	1178.9188	Y
75.81375	Y	1007.5262	Y	120.72	Y	227.70625	Y
80.73875	Y	2076.9025	Y	93.3575	Y	1813.4562	Y
170.11875	Y	737.695	Y	270.19625	Y	75.95	Y
62.7375	Y	148.04125	Y	599.6975	Y	247.52625	Y
390.31125	Y	222.55875	Y	391.355	Y	249.15125	Y
117.185	Y	104.78125	Y	124.91875	Y	210.90125	Y
519.08125	Y	175.55125	Y				

Table 14.2. Sorted expression level of gene IL-8 in 22 normal and 40 colon cancer tissues used in Fig. 14.1

Expression level	Colon cancer	Expression level	Colon cancer	Expression level	Colon cancer	Expression level	Colon cancer
11.37625	N	23.74	N	28.00875	N	28.7675	N
30.57875	N	31.6975	N	33.9725	N	35.95875	N
36.8675	N	39.7575	N	40.33875	N	45.1	N
55.2	N	56.91875	N	62.7375	Y	75.81375	Y
75.95	Y	76.125	Y	76.9875	N	80.73875	Y
93.3575	Y	97.92	N	104.78125	Y	113.91375	N
114.92375	Y	117.185	Y	117.6025	N	120.72	Y
124.91875	Y	131.65875	Y	148.04125	Y	169.1375	Y
170.11875	Y	171.4525	N	175.55125	Y	186.2775	Y
210.90125	Y	213.6275	Y	222.55875	Y	227.70625	Y
238.58625	N	247.52625	Y	249.15125	Y	270.19625	Y
284.14625	Y	645.99375	N	306.30875	Y	311.4375	Y
326.42625	Y	370.04	Y	390.31125	Y	391.355	Y
412.135	Y	519.08125	Y	567.13125	N	599.6975	Y
737.695	Y	1007.5262	Y	1178.9188	Y	1528.4062	Y
1813.4562	Y	2076.9025	Y				

is only a finite number of them that make a difference. In other words, it takes a finite number of steps to find an optimal threshold, although the solution is not unique.

The first step in finding an optimal threshold is to sort all expression levels, say, in an ascending order as displayed in Table 14.2. If the threshold is below the minimum (11.37625) or above the maximum (2076.9025), it produces an empty

subsample. Thus, the threshold should be between 11.37625 and 2076.9025. If we take a look at the two lowest levels, 11.37625 and 23.74, it is clear that any threshold between these two levels produces the same two subsamples (or daughter nodes). In this example, there are 62 distinct levels of expression. Thus, we have $62 - 1 = 61$ distinct ways to split the 62 samples into two daughter nodes. It is noteworthy that, unlike this example, the number of unique levels of a predictor is usually lower than the number of samples.

The second step in finding an optimal threshold is to move along the intervals defined by two adjacent, distinct levels of the sorted predictor values. In Table 14.2, we move along as follows:

$$[11.37625, 23.74), [23.74, 28.00875), \dots, [56.91875, 62.7375), \\ \dots, [1528.4062, 1813.4562), [1813.4562, 2076.9025).$$

For computation, the threshold can be chosen as the middle point of the above intervals. For interpretation, the threshold can be rounded-off as is done to the first split in Fig. 14.1.

We have determined the pool of the potential thresholds, which is sometimes referred to as the allowable splits. Obviously, we can examine each threshold one at a time and assess its quality according to (14.1).

Table 14.3. Search for the optimal split

Interval	Left node			Right node			Split Quality g_s
	No. of sample	No. of cancer	Node impurity	No. of sample	No. of cancer	Node impurity	
[11.37625, 23.74)	1	0	0	61	40	0.6438	0.3666
[23.74, 28.00875)	2	0	0	60	40	0.6365	0.3849
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
[56.91875, 62.7375)	14	0	0	48	40	0.4506	0.6512
[62.7375, 75.81375)	15	1	0.1030	47	39	0.4562	0.6292
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
[1528.4062, 1813.4562)	60	38	0.6572	2	2	0	0.3640
[1813.4562, 2076.9025)	61	39	0.6538	1	1	0	0.3568

For a large data set, this means a lot of wasted computing time. To reduce the computation to a minimal level, let us take a careful look as to what happens when we move the threshold from one interval to the next. In Table 14.3, as the threshold is moved up to the next interval, the samples that were already assigned to the left daughter stay on the left side because their expression levels are still below the new threshold. Most of the samples that were assigned to the right daughter stay on the right side, except those samples whose expression levels are equal to the lower limit of the new interval. In this particular case, there is only one sample that we need to move from the right side to the left every time we move the threshold by one interval. This observation implies that the node impurities and the split

quality can be computed by updating the information slightly for the small set of the samples that are affected. Every of such a small set of samples is affected only once in the entire search of the predictor. In summary, after the values of a predictor are sorted, we can find an optimal threshold to split a node in the number of steps proportional to the number of distinct values of the predictor, which is at most the number of samples in the node. For the present example, any value in $[56.91875, 62.7375)$ is an optimal split. Intuitively from Table 14.3, we push the threshold as high as possible to maintain the perfect purity of the left daughter node. In the meantime, if we look bottom-up from the table, we also push the threshold as low as possible to maximize the purity of the right daughter node. The interval $[56.91875, 62.7375)$ offers the best balance. In Fig. 14.1, the split is chosen at 60, although any number in this interval is a legitimate choice.

Overall, if we have n samples in a node and p predictors, excluding the sorting time, the final threshold for the node can be identified in at most $O(np)$ steps.

Splits Based on a Nominal Predictor

14.3.2

For a nominal variable, we cannot sort the values of the variable as we did in Table 14.2. For a predictor of k levels, there are a total of $2^{k-1} - 1$ ways to split a node. To explain the algorithm, let us use an artificial example as summarized in Table 14.4.

Table 14.4. An artificial data set

Predictor value	No. of normal	No. of cancer	Rate of cancer
A	5	10	0.67
B	10	5	0.33
C	20	30	0.60
D	35	25	0.42

In Table 14.4, the predictor has 4 levels, giving rise to 7 possible ways to split a node. A naive way is to assess every allowable split on an individual basis. This could be an extensive computation when the number of levels is 10 or higher. Thus, it is important to find a way to compute the quality of all splits in a gradual manner as in Sect. 14.3.1. If we focus on the levels of the predictor for the left daughter node, we can travel all 7 possible splits as follows: $\{A\}$, $\{AB\}$, $\{B\}$, $\{BC\}$, $\{C\}$, $\{AC\}$, and $\{ABC\}$. The key is that every move requires either the deletion or addition of a single level, which keeps the computation at the minimal level. Such a path of traveling through all $2^{k-1} - 1$ splits can be defined for any k .

There is actually a simple and quick solution for a dichotomous response. As shown in Table 14.4, we can compute the cancer rate for every level of the nominal predictor. During the splitting, the rates can substitute for the corresponding nominal levels. Because the rates are ordinal, the method described in Sect. 14.3.1 can be applied. After the optimal split is determined, we can map the rate back

to the original nominal level. For example, for the data in Table 14.4, the optimal threshold based on the rate is in the interval $[0.42, 0.6)$, which means that the left daughter node contains samples with levels B and D, and the right daughter node with levels A and C. For a multiclass response, there is no apparent way to form an ordinal surrogate for a nominal predictor.

14.3.3 Missing Values

An important feature of decision trees is their ability to deal with missing predictor values. There are several solutions. Although there have been limited attempts (Quinlan, 1989) to compare some of them, the performance of the various solutions is largely unexplored. The choice mostly depends on the objective of the study.

The easiest approach is to treat the missing attribute as a distinct value and to assign all samples with missing values to the same node (Zhang et al., 1996). This approach is not only simple, but also provides clear paths as to where the samples with missing attributes end up in the tree structure.

Breiman et al. (1984) introduced and advocated surrogate splits to deal with missing attributes. The idea is very intuitive. For example, in Table 14.2, we considered using expression levels from gene IL-8 to split the 62 samples. What happens if the expression level from one of the samples, say, the first one, was not recorded? This happens in microarray experiments. Because IL-8 level is missing for the first sample, we cannot determine whether the level is below or above 60 and hence cannot decide whether the first sample should be assigned to the left or right daughter node. To resolve this ambiguity, Breiman et al. (1984) proposed to seek help from other genes that act “similarly” to IL-8. Since there are many other genes, we can use the one that is most similar to IL-8, which leads to a surrogate for IL-8.

What we need to clarify is the meaning of similarity. To illustrate this concept, let us consider gene CANX. Using the method described in Sect. 14.3.1, we can find an optimal split from gene CANX. The similarity between CANX and IL-8 is the probability that the optimal splits from these two genes assign a sample with complete information in these two genes into the same node. This strategy is similar to replacing a missing value in one variable in linear regression by regressing on the non-missing value most highly correlated with it. Then, why can't we use the same strategy as in the linear regression? According to Breiman et al. (1984), their strategy is more robust. The main reason is that their strategy is more specific to the particular sample with missing attributes, and does not result in a potential catastrophic impact for other samples with missing attributes.

The surrogate splits have some advantages over the simpler approach as described earlier. It makes use of other potentially useful information. Breiman et al. (1984) also proposed to rank the importance of variables through surrogate splits. The surrogate splits also have some limitations. First, it is uncommon, if at all, that surrogate splits are provided in published applications. Thus, it is unrealistic to know what the surrogate splits are and how we assign a sample with a missing attribute. Second, there is no guarantee in a data set that we can find a satisfactory surrogate split. Lastly, while it is a sensible idea to rank the variable importance

based on surrogate splits, there is no assurance that a predictor ranked relatively high is necessarily predictive of the outcome, which can create a dilemma for interpretation. More recently, the importance of a variable tends to be evaluated on the basis of its performance in forests (Breiman, 1994; Zhang et al., 2003) rather than on a single tree.

In the construction of random forests, Breiman proposed another way of replacing missing values through an iterative process. A similar idea can be applied for tree construction. To initialize the process, we can fill in the missing values by the median of an ordered variable or by the category of a nominal variable with the highest frequency. An initial tree can be constructed once all missing data are imputed. In the next step, suppose again that in Table 14.2, the expression of gene IL-8 is missing for the first sample. The unobserved level is estimated by a weighted average over the samples with observed expressions for gene IL-8. Here, the weight is the so-called proximity, which is a similarity measure between a pair of samples. Intuitively, if the second sample is more similar to the first sample than to the third one, we give more weight to the second sample than to the third one if the first sample is not observed. How is the proximity defined for a pair of samples? We can set it to zero before the initial tree is grown. Then, whenever a tree is grown, if two samples end up in the same terminal nodes, its proximity is increased by one unit. After the missing data are updated, a new tree is grown. Breiman recommends to continue this process at most five times in the random forest construction. For tree construction, it may take longer for the process to “converge”, especially when the number of predictors is large. Nonetheless, it may still be worthwhile to repeat a few iterations. In addition to this convergence issue, it is also difficult to track where the samples with missing values are assigned as with the use of surrogate splits.

Interpretation

Interpretation of results from trees is usually straightforward. In Fig. 14.1, we identified 3 genes IL-8, CANX, and RAB3B whose expression levels are highly predictive of colon cancer. However, this does not necessarily mean that these genes cause colon cancer. Such a conclusion requires a thorough search of the literature and further experiments. For example, after reviewing the literature, Zhang et al. (2001) found evidence that associates IL-8 with the stage of colon cancer (Fox et al., 1998), the migration of human clonic epithelial cell lines (Toshina et al., 2000), and metastasis of bladder cancer (Inoue et al., 2000). In addition, the expression of the molecular chaperone CANX was found to be down-regulated in HT-29 human colon adenocarcinoma cells (Yeates and Powis, 1997) and to be involved in apoptosis in human prostate epithelial tumor cells (Nagata et al., 1997). Lastly, RAB3B is a member of the RAS oncogene family. Therefore, these existing studies provide independent support that the three genes identified in Fig. 14.1 may be in the pathways of colon cancer. If this hypothesis could be confirmed from further

experiments, Fig. 14.1 would have another important implication. Pathologically speaking, the 40 colon cancer samples are indistinguishable. Figure 14.1 indicates that those 40 samples are not homogeneous in terms of gene expression levels. If confirmed, such a finding could be useful in cancer diagnosis and treatment.

As we stated earlier, there are numerous applications of decision trees in biomedical research, including the example above. To have a glimpse of the diverse applications of decision trees, let us review two different examples.

-
- 3** **Example 3** Frydman and colleagues introduced recursive partitioning for financial classification (Frydman et al., 2002). They considered a financial dataset of 58 bankrupt ($y = 1$) industrial companies that failed during 1971–81, and 142 non-bankrupt ($y = 0$) manufacturing and retailing companies randomly selected from COMPUSTAT universe. Each company forms an observational unit or the so-called sample. Twenty financial variables with prior evidence of predicting business failure are considered. They include the ratio of cash to total assets, the ratio of cash to total sales, the ratio of cash flow to total debt, the ratio of current assets to current liabilities, the ratio of current assets to total assets, the ratio of current assets to total sales, the ratio of earnings before interest and taxes to total assets, interest coverage, the ratio of market value of equity to total capitalization, the ratio of net income to total assets, the ratio of quick assets to current liabilities, the ratio of quick assets to total assets, the ratio of quick assets to total sales, the ratio of retained earnings to total assets, the ratio of total debt to total assets, the ratio of total sales to total assets, and the ratio of working capital to total sales.

We can see from Fig. 14.2 that the risk of bankruptcy is relatively high if the ratio of cash flow to total debt is below 0.1309, unless both the ratio of retained earnings to total assets and the ratio of cash to total sales are above certain levels, i.e., 0.1453 and 0.025, respectively. Even if the ratio of cash flow to total debt is above 0.1309, there can be elevated risk of bankruptcy if the ratio of total debt to total assets is high (above 0.6975). A tree diagram as in Fig. 14.2 offers a very clear and simple assessment of the financial state of a company.

-
- 2** **Example 2:** (*continued*) We indicated earlier what the predictors and response are for Example 2. Let us revisit this example. Unlike the other examples that we have introduced so far, this example uses a continuous response y – the compound potency. Because of this difference, the resulting tree is called a regression tree. To utilize the information from the 3-dimensional structures of compounds, Chen et al. (1998) used atom pair descriptors that are composed of the atom types of the two component atoms and the “binned” Euclidean distance between these two atoms. The width of each distance bin was chosen as 1.0 Å. To define predictors x from the atom pair descriptors, the authors characterized the atom pair descriptors in 17 types including negative charge center (e.g., sulfinic group), positive charge center (e.g., the nitrogen in primary, secondary, and tertiary amines), hydrogen

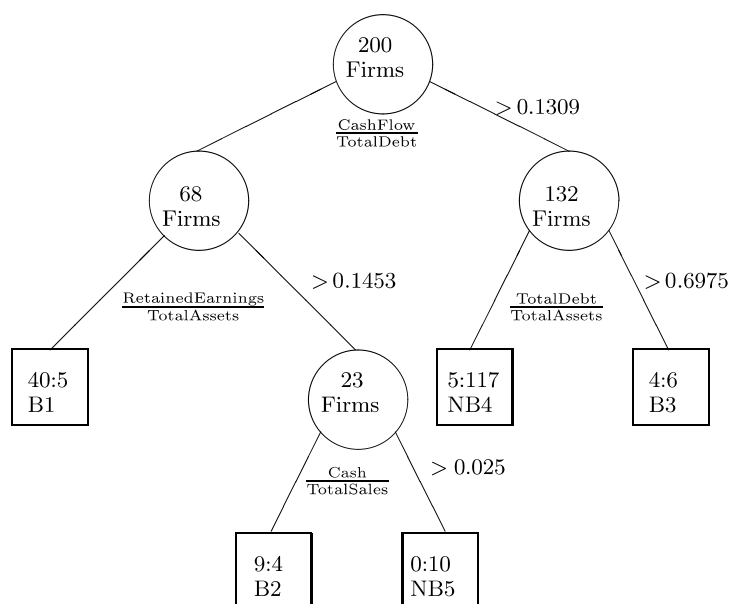


Figure 14.2. Classification tree for bankruptcy. B1, B2, and B3 are three groups of relatively high risk of bankruptcy, and NB1 and NB2 are two groups of likely non-bankrupt companies. Inside the terminal nodes (*boxes*) are the numbers of bankrupt and non-bankrupt companies. See Frydman et al. (2002) for more details

bond acceptor (e.g., oxygen with at least one available lone pair electron), triple bond center, aromatic ring center, and H-bond donor hydrogen.

Figure 14.3 presents part of the regression tree that is constructed by Chen et al. (1998). We trimmed the left hand side to fit into the space here; however, we can get the idea from the right hand side of tree. Generally speaking, a node of size 3 or 6 such as nodes 6 and 8 is too small to be reliable. Since we do not have the data to re-grow the tree, let us pretend that the node sizes are adequate, and concentrate on the interpretation instead. Since the main objective of Chen et al. appears to identify active nodes (i.e., those with high potencies), a small, inactive node is not of great concern.

First, there is one highly active node (node 7 with potency greater than 2) in Fig. 14.3. There are also two highly active nodes on the left hand side which are not shown in Fig. 14.3. Supported by the literature, Chen et al. (1998) postulated that there might be different mechanisms of action because the active nodes contain compounds of very different characteristics. This is similar to the hypothesis suggested by Fig. 14.1 that the 40 colon cancer tissues might be biologically heterogeneous. Chen et al. concluded further that their tree demonstrates the ability to detect multiple mechanisms of action coexisting in a large three-dimensional chemical data set. In addition, the selected atom pair descriptors also reveal interesting features of the monoamine oxidase (MAO) inhibitors. For instance, the

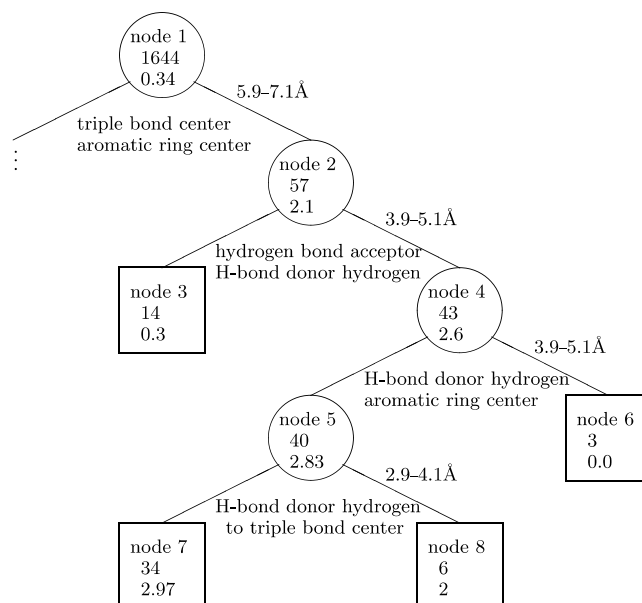


Figure 14.3. Regression tree for predicting potencies of compounds. Inside each node are the number of compounds (*middle*) and the average potency of all compounds within the node (*bottom*).

Underneath each node is the selected atom pair descriptor. Above the arm is the interval for the distance between the selected atom pair descriptor that assigns the compounds to the right daughter node. See Chen et al. (1998) for more details

“aromatic ring center–triple bond center” pair in the first split is the structural characteristic of pargyline, a well known MAO inhibitor.

We can see from these examples that tree-based methods tend to unravel integrated, intuitive results whose pieces are consistent with prior findings. Not only can we use trees for prediction, but also we may use them to identify potentially important mechanisms or pathways for further investigation.

14.5

Survival Trees

The most popular use of tree-based methods is arguably in survival analysis for censored time, particularly in biomedical applications. The general goal of such applications is to identify prognostic factors that are predictive of survival outcome and time to an event of interest. For example, Banerjee et al. (2000) reported a tree-based analysis that enables the natural identification of prognostic groups among patients in the perioperative phase, using information available

regarding several clinicopathologic variables. Such groupings are important because patients treated with radical prostatectomy for clinically localized prostate carcinoma present considerable heterogeneity in terms of disease-free survival outcome, and the groupings allow physicians to make early yet prudent decisions regarding adjuvant combination therapies. See, e.g., Bacchetti and Segal (1995), Carmelli et al. (1991), Carmelli et al. (1997) and Kwak et al. (1990) for additional examples.

Before pointing out the methodological challenge in extending the basic classification trees to survival trees, let us quickly introduce the censored data. Let z denote the time to an event, which can be death or the occurrence of a disease. For a variety of reasons including losts to follow-up and the limited period of a study, we may not be able to observe z until the event occurs for everyone in the study. Thus, what we actually observe is a censored time y which is smaller than or equal to z . When z is observed, $y = z$. Otherwise, z is censored and $y < z$. Let $\delta = 1$ or 0 denote whether z is censored or observed.

The question is how to facilitate the censored time y in the tree-based methods. As in Sect. 14.2, we need to define a splitting criterion to divide a node into two, and also to find a way to choose a “right-sized” tree. Many authors have proposed different methods to address these needs. Here, we describe some of the methods. See Crowley et al. (1995), Intrator and Kooperberg (1995), LeBlanc and Crowley (1995), Segal (1988), Segal (1995), Zhang et al. (2001) and Zhang and Singer (1999) for more details.

Maximizing Difference Between Nodes

14.5.1

Gordon and Olshen (1985) are among the earliest to have developed survival trees. Earlier, we focused on reducing the impurity within a node by splitting. When two daughter nodes have low impurities, the distributions of the response tend to differ between the two nodes. In other words, we could have achieved the same goal by maximizing the difference between the distributions of the response in the two daughter nodes. There are well established statistics that measure the difference in distribution. In survival analysis, we can compute the Kaplan–Meier curves (see, e.g., Miller, 1981) separately for each node. Gordon and Olshen used the so-called L^p Wasserstein metrics, $d_p(\cdot, \cdot)$, as the measure of discrepancy between the two survival functions. Specifically, for $p = 1$, the Wasserstein distance, $d_1(S_L, S_R)$, between two Kaplan–Meier curves, S_L and S_R , is illustrated in Fig. 14.4.

A desirable split maximizes the distance, $d_1(S_L, S_R)$, where S_L and S_R are the Kaplan–Meier curves for the left and right daughter nodes, respectively. Replacing g_s in (14.1) with $-d_1(S_L, S_R)$ we can split the root node into two daughter nodes and use the same recursive partitioning process as before to produce a saturated tree.

To prune a saturated survival tree, T , Gordon and Olshen (1985) generalized the tree cost-complexity for censored data. The complexity remains the same as before, but we need to redefine the cost $R(t)$, which now is measured by how far node t deviates from a desirable node in lieu of a pure node in the binary response case. In the present situation, a replacement for a pure node is a node τ in which all

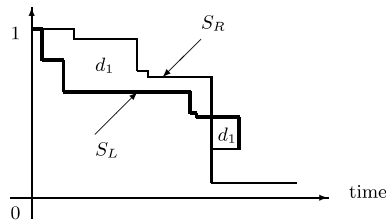


Figure 14.4. The L^1 Wasserstein distance between two Kaplan–Meier curves as measured by the area marked with d_1 . Note that one curve (S_L) is thicker than the other (S_R)

observed times are the same, and hence its Kaplan–Meier curve, δ_τ , is a piecewise constant survival function that has at most one point of discontinuity. Then, the within-node cost, $R(t)$, is defined as $d_1(S_t, \delta_\tau)$. Combining this newly defined cost-complexity with the previously described pruning step serves as a method for pruning survival trees.

Another, perhaps more commonly used way to measure the difference in survival distributions is to make use of the log-rank statistic. Indeed, the procedures proposed by Ciampi et al. (1986) and Segal (1988) maximize the log-rank statistic by comparing the survival distributions between the two daughter nodes. The authors did not define the cost-complexity using the log-rank statistic. However, LeBlanc and Crowley (1993) introduced the notion of “goodness-of-split” complexity as a substitute for cost-complexity in pruning survival trees. Let $G(t)$ be the value of the log-rank test at node t . Then the split-complexity measure is

$$G(T) = \sum_{t \notin \tilde{T}} G(t) - \alpha (|\tilde{T}| - 1) .$$

Therneau et al. (1990) proposed another way to define $R(t)$ that makes use of the so-called martingale residuals by assuming within-node proportional hazard models and then the least squares are computed as the cost.

In our experience, we found that Segal’s bottom-up procedure (Segal, 1988) is practical and easy to use. That is, for each internal node (including the root node) of a saturated tree, we assign it a value that equals the maximum of the log-rank statistics over all splits starting from the internal node of interest. Then, we plot the values for all internal nodes in an increasing order and decide a threshold from the graph. If an internal node corresponds to a smaller value than the threshold, we prune all of its offspring. Zhang and Singer (1999) pointed out that this practical procedure can be modified in a broad context by replacing the log-rank statistic with a test statistic that is appropriate for comparing two samples with a defined outcome.

14.5.2 Use of Likelihood Functions

Although the concept of node impurity is very useful in the development of tree-based methodology, that concept is closely related to the concept of likelihood as

pointed out by Zhang et al. (2001). In fact, the adoption of likelihood makes it much easier to extend the tree-based methodology to analysis of complex dependent variables including censored time. For example, Davis and Anderson (1989) assume that the survival function within any given node is an exponential function with a constant hazard. LeBlanc and Crowley (1992) and Ciampi et al. (1988) assume different within-node hazard functions. Specifically, the hazard functions in two daughter nodes are assumed proportional, but are unknown. In terms of estimation, LeBlanc and Crowley (1992) use the full or partial likelihood function in the Cox proportional hazard model whereas Ciampi et al. (1988) use a partial likelihood function.

The most critical idea in using the likelihood is that within-node survival functions are temporarily assumed to serve as a vehicle of finding a split instead of believing them to be the true ones. For example, we cannot have a constant hazard function in the left daughter node, and then another constant hazard function in the right daughter node while assuming that the parent node also has a constant hazard function. Here, the constant hazard function plays the role of the “sample average”. However, after a tree is constructed, it is both reasonable and possible that the hazard functions within the terminal nodes may become approximately constant.

A Straightforward Extension

14.5.3

Zhang (1995) examined a straightforward tree-based approach to censored survival data by observing the fact that the response variable involves two dimensions: a binary censoring indicator and the observed time. If we can split a node so that the node impurity is “minimized” in both dimensions, the within-node survival distribution is expected to be homogeneous. Based on this intuitive idea, Zhang (1995) proposed to compute the within-node impurity in terms of both the censoring indicator and the observed time first separately, and then together through weighting. Empirically, this simple approach tends to produce trees similar to those produced from using the log-rank test. More interestingly, empirical evidence also suggests that this simple approach outperforms its more sophisticated counterparts in discovering the underlying structures of data. Unfortunately, there need to be more comparative studies to scrutinize these different methods, even though limited simulations comparing some of the methods have been reported in the literature (Crowley et al., 1995; Crowley et al., 1997; Zhang, 1995).

Other Developments

14.5.4

The methods that we described above are not designed to deal with time-dependent covariates. Bacchetti and Segal (1995) and Huang et al. (1998) proposed similar approaches to accommodate the time-dependent covariates in survival trees. The main concern with these existing approaches is that the same subject can be assigned to both the left and right daughter nodes, which is distinct from any other tree-based methods and is potentially confusing in interpretation.

It is common in survival tree analysis that we want to stratify our sample into a few groups that define the grades for the survival. To this end, it is useful to combine some terminal nodes into one group, which is loosely called “amalgamation”. Ciampi et al. (1986) used the log-rank statistic for this purpose. LeBlanc and Crowley (1993) proposed constructing an ordinal variable that describes the terminal nodes. Often, we can simply examine the Kaplan–Meier curves for all terminal nodes to determine the group membership (Carmelli et al., 1997).

Tree-based Methods for Multiple Correlated Outcomes

14.6

As pointed out by Zhang (1998), multiple binary responses arise from many applications for which an array of health-related symptoms are of primary interest. Most of the existing methods are parametric; see, e.g., Diggle et al. (1994) for an excellent overview. In this section, we will describe a tree-based alternative to the parametric methods.

Motivated by both the broad application as well as by the need to analyze building-related occupant complaint syndrome (BROCS), Zhang (1998) proposed a tree-based method to model and classify multiple binary responses. Let us use the BROCS study to explain the method.

To understand the nature of BROCS, data were collected in 1989 from 6800 employees of the Library of Congress (LOC) and the headquarters of the Environmental Protection Agency (EPA) in the United States. The data contain many explanatory variables, but Zhang (1998) extracted a subset of 22 putative risk factors, most of which are answers to “yes or no” or frequency (never, rarely, sometimes, etc.) questions. For example, is working space an enclosed office with door, a cubicle without door, stacks, etc? See Table 1 of Zhang (1998) for a detailed list. In this data set, BROCS is represented by six binary responses that cover respiratory symptoms in the central nervous system, upper airway, pain, flu-like, eyes, and lower airway. The primary purpose with this extracted data set is to evaluate the effect of the important risk factors on the six responses by constructing trees.

In terms of notation, the primary distinction is that the response y for each subject is a 6-vector. Consequently, we need to generalize the node-splitting criterion and cost-complexity to this vector-response. As we indicated earlier, one solution is to assume a certain type of within-node distribution for the vector-response and then maximize the within-node likelihood for splitting. One such distribution is

$$f(y; \Psi, \theta) = \exp(\Psi'y + \theta'w - A(\Psi, \theta)) , \quad (14.4)$$

where Ψ and θ are node-dependent parameters, $A(\Psi, \theta)$ is the normalization function depending on Ψ and θ , and $w = \sum_{i < j} y_i y_j$. Zhang (1998) chose this distribution because it is commonly used in the parametric models for multiple binary responses. See, e.g., Cox (1972), Fitzmaurice and Laird (1993) and Zhao and

Prentice (1990). The negative of the likelihood based on (14.4) now serves as the impurity function, and the rest of the recursive partitioning as described before applies.

A naive approach is to treat y as a numerical vector and use a function such as the determinant of the within-node covariance matrix of y as a measure of impurity. If y were continuous, this approach is what Segal (1992) proposed to construct regression trees for repeatedly measured continuous y . For binary outcomes, however, this approach appears to suffer the well-known end-cut preference problem in the sense that it gives preference to the splits that result in two unbalanced daughter nodes in terms of their sizes.

One advantage of the likelihood based method is that the negative of the within-node likelihood can also be used as the within-node cost $R(t)$ for tree pruning. The main difficulty with this method is the computational burden, because every allowable split calls for a maximization of the likelihood derived from (14.4). Some strategies for reducing the computational time are discussed in Zhang (1998).

The criterion based on (14.4) ultimately leads to a 9 terminal nodes tree as displayed in Fig. 14.5, which suggests that respondents belonging to terminal nodes 7 and 17 have high incidence of respiratory symptoms. This is because the working area air quality of the people within these terminal nodes was poor, namely, often too stuffy or sometimes dusty. On the other hand, for example, subjects in terminal node 14 experienced the least discomfort because they had the best air quality. The basic message from this example is that tree-based analyses often reveal findings that are readily interpretable.

Remarks

14.7

In Breiman et al. (1984), tree-based methods are presented primarily as an automated machine learning technique. There is now growing interest in applying tree-based methods in biomedical applications, partly due to the rising challenges in analyzing genomic data in which we have a large number of predictors and a far smaller number of observations (Zhang et al., 2001). In biomedical applications, scientific understanding and interpretation of a fitted model are an integral part of the learning process. In most situations, an automated tree as a whole has characteristics that are difficult or awkward to interpret. Thus, the most effective and productive way of conducting tree-based analyses is to transform this machine learning technique into a human learning technology. This requires the users to review the computer-generated trees carefully and revise the trees using their knowledge, which not only often simplifies the trees, but also may improve the predictive precision of the trees, because recursive partitioning is not a forward looking process and does not guarantee any optimality of the overall tree. Zhang et al. (1996) called this step tree repairing.

While the full potential of tree-based applications remains to be seen and exploited, it must be made crystally clear that parametric methods such as logistic

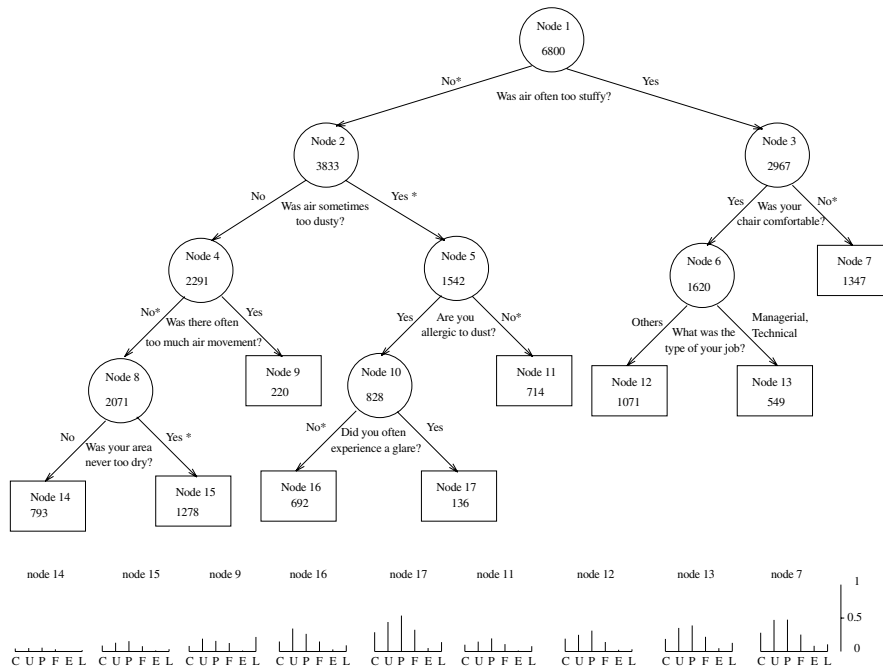


Figure 14.5. Tree structure for the risk factors of BROCS based on (14.4). Inside each node (a circle or a box) are the node number and the number of subjects. The splitting question is given under the node. The asterisks indicate where the subjects with missing information are assigned. The pin diagrams under the tree show the incidence rates of the six clusters (C: CNS; U: upper airway; P: pain; F: flu-like; E: eyes; and L: lower airway) in the nine terminal nodes. The side bar on the right end indicates the range of 0 and 1 for the rates of all symptoms

regression and Cox models will continue to be useful statistical tools. We will see more applications that use tree-based methods together with parametric methods to take advantages of various types of methods. The main advantage of tree-based methods is their flexibility and intuitive structures. However, because of their adaptive nature, statistical inference based on tree-based methodology is generally difficult. Despite the difficulty, some progress has been made to understand the asymptotic behavior of tree-based inference (Breiman, 1994; Buhlmann and Yu, 2003; Donoho, 1997; Gordon and Olshen, 1978; Gordon and Olshen, 1980; Gordon and Olshen, 1984; Lugosi and Nobel, 1996; Nobel, 1996; Nobel and Olshen, 1996).

Some attempts have been made to compare the tree-structured methods with other methods (Long et al., 1993; Segal and Bloch, 1989; Selker et al., 1995). More comparisons are still warranted, particularly in the context of genomic applications where data reduction is necessary and statistical inference is also desirable.

One exciting development in recent years is the expansion of trees into forests. In a typical application such as Banerjee et al. (2000) and Carmelli et al. (1997),

constructing one or several trees is usually sufficient to unravel relationships between predictors and a response. Nowadays, many studies produce massive information such as recognizing spam mail from numerous characteristics and identifying disease genes. One or even several trees are no longer adequate to convey all of the critical information in the data. Construction of forests enables us to discover data structures further and in the meantime improves classification and predictive precision (Breiman, 1994; Zhang et al., 2003). So far, most forests are formed through some random perturbations and are hence referred to as random forests (Breiman, 1994). For example, we can draw bootstrap samples (Efron and Tibshirani, 1993) from the original sample and construct a tree as described above. Every time we draw a bootstrap sample, we produce a tree. Repetition of this process yields a forest. This is commonly called bagging (Breiman, 1994). The emergence of genomic and proteomic data afford us the opportunity to construct deterministic forest (Zhang et al., 2003) by collecting a series of trees that have a similarly high predictive quality. Not only do forests reveal more information from large data sets, but they also outperform single trees (Breiman, 1994; Buhlmann and Yu, 2002; Buhlmann and Yu, 2003; Zhang et al., 2003).

A by-product of forests is a collection of variables that are frequently used in the forests, and the frequent uses are indicative of the importance of those variables. Zhang et al. (2003) examined the frequencies of the variables in a forest and used them to rank the variables. It would be even more helpful and informative if a certain probability measure could be assigned to the ranked the variables.

Bayesian approaches may offer another way to construct forests by including trees with a certain level of posterior probability. These approaches may also help us understand the theoretical properties of tree-based methods. However, the existing Bayesian tree framework focuses on providing an alternative method to those that exist. We would make an important progress if we could take full advantage of the Bayesian approach to improve our tree-based inference.

Classification and regression trees assign a subject to a particular node following a series of boolean statements. Ciampi et al. (2002) considered a “soft” splitting algorithm that at each node an individual goes to the right daughter node with a certain probability, which is a function of a predictor. This approach has the spirit of random forests. In fact, we can construct a random forest by repeating this classification scheme.

Several companies including DTREG.com, Insightful, Palisade Corporation, Salford Systems, and SAS market different variants of decision trees. In addition, there are many versions of free-ware including my own version, which is distributed from my website.

Acknowledgements. This work is supported in part by grants R01DA12468, R01DA16750 and K02DA017713 from the National Institutes of Health. The author wishes to thank Norman Silliker, Elizabeth Triche, Yuanqing Ye and Yinghua Wu for their helpful assistance.

References

- Altman, E.I. (2002). *Bankruptcy, Credit Risk and High Yield Junk Bonds*. Blackwell Publishers, Malden, MA.
- Alon, U., Barkai, N., Notterman, D.A., Gish, K., Ybarra, S., Mack, D. and Levine, A.J. (1999). Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays. *Proceedings of the National Academy of Sciences*, 96: 6745–6750.
- Bacchetti, P. and Segal, M.R. (1995). Survival trees with time-dependent covariates: application to estimating changes in the incubation period of AIDS. *Lifetime Data Analysis*, 1: 35–47.
- Bahl, L.R., Brown, P.F., de Sousa, P.V., and Mercer R.L. (1989). A tree-based language model for natural language speech recognition. *IEEE Trans. on AS and SP*, 37: 1001–1008.
- Banerjee, M., Biswas, D., Sakr, W., and Wood, D.P. Jr. (2000). Recursive partitioning for prognostic grouping of patients with clinically localized prostate carcinoma. *Cancer*, 89: 404–411.
- Breiman, L., Friedman, J.H., Olshen, R.A. and Stone, C.J. (1984). *Classification and Regression Trees*, Wadsworth, Belmont, California.
- Breiman, L. (1994). Bagging predictors. *Machine Learning*, 26: 123–140.
- Brennan, N., Parameswaran, P. et al. (2001). *A Method for Selecting Stocks within Sectors*, Schroder Salomon Smith Barney.
- Buhlmann, P. and Yu, B. (2003). Boosting with the L-2 loss: Regression and classification. *Journal of the American Statistical Association*, 98: 324–339.
- Buhlmann, P. and Yu, B. (2002). Analyzing bagging. *Annals of Statistics*, 30: 927–961.
- Buntine, W. and Niblett, T. (1992). A further comparison of splitting rules for decision-tree induction. *Machine Learning*, 8: 75–85.
- Carmelli, D., Halpern, J., Swan, G.E., Dame, A., McElroy, M., Gelb, A.B., and Rosenman, R.H. (1991). 27-year mortality in the western collaborative group study: construction of risk groups by recursive partitioning. *Journal of Clinical Epidemiology*, 44: 1341–1351.
- Carmelli, D., Zhang, H.P. and Swan, G.E. (1997). Obesity and 33 years of coronary heart disease and cancer mortality in the western collaborative group study. *Epidemiology*, 8: 378–383.
- Carter, C. and Catlett, J. (1987). Assessing credit card applications using machine learning. *IEEE Expert*, 2: 71–79.
- Chen, X., Rusinko, A. and Young, S.S. (1998). Recursive partitioning analysis of a large structure-activity data set using three-dimensional descriptors. *Journal of Chemical Information and Computer Sciences*, 38: 1054–1062.
- Ciampi, A., Couturier, A. and Li, S.L. (2002). Prediction trees with soft nodes for binary outcomes. *Statistics in Medicine*, 21: 1145–1165.

- Ciampi, A., Hogg, S., McKinney, S. and Thiffault, J. (1988). A computer program for recursive partition and amalgamation for censored survival data. *Computer Methods and Programs in Biomedicine*, 26: 239–256.
- Ciampi, A., Thiffault, J., Nakache J.-P. and Asselain, B. (1986). Stratification by stepwise regression, correspondence analysis and recursive partition: A comparison of three methods of analysis for survival data with covariates. *Computational Statistics and Data Analysis*, 4: 185–204.
- Cox, D.R. (1972). The analysis of multivariate binary data. *Applied Statistics*, 21: 113–120.
- Cristianini, N. and Shawe-Taylor, J. (2000). *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*, Cambridge University Press, Cambridge.
- Crowley, J., LeBlanc, M., Gentleman, R. and Salmon, S. (1995). Exploratory methods in survival analysis. In Koul, H.L. and Deshpande, J.V. (eds), *IMS Lecture Notes – Monograph Series 27*, pp.55–77, IMS, Hayward, CA.
- Crowley, J., LeBlanc, M., Jacobson, J. and Salmon S. (1997). Some exploratory methods for survival data. In Lin, D.Y. and Fleming, T.R. (eds), *Proceedings of the First Seattle Symposium in Biostatistics*, Springer, New York.
- Davis, R. and Anderson, J. (1989). Exponential survival trees. *Statistics in Medicine*, 8: 947–962.
- Desilva, G.L. and Hull, J.J. (1994). Proper noun detection in document images. *Pattern Recognition*, 27: 311–320.
- Diggle, P.J., Liang, K.Y. and Zeger, S.L. (1994). *Analysis of Longitudinal Data*, Oxford Science Publications, New York.
- Donoho, D.L. (1997). CART and best-ortho-basis: A connection. *Annals of Statistics*, 25: 1870–1911.
- Efron, B. and Tibshirani, R.J. (1993). *An Introduction to the Bootstrap*, Chapman & Hall, New York.
- Fisher, R.A. (1936). The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7: 179–188.
- Fitzmaurice, G. and Laird, N.M. (1993). A likelihood-based method for analyzing longitudinal binary responses. *Biometrika*, 80: 141–151.
- Fox, S.H., Whalen, G.F., Sanders, M.M., Burlison, J.A., Jennings, K., Kurtzman, S. and Kreutzer, D. (1998). Angiogenesis in normal tissue adjacent to colon cancer. *Journal of Surgical Oncology*, 69: 230–234.
- Friedman, J.H. (1977). A recursive partitioning decision rule for nonparametric classification. *IEEE Trans. Computers.*, C-26: 404–407.
- Frydman, H., Altman, E.I. and Kao, D.-I. (2002). Introducing Recursive Partitioning for Financial Classification: The Case of Financial Distress. In *Altman ed. Bankruptcy, Credit Risk and High Yield Junk Bonds*, pp.37–59.
- Geman, D. and Jedynek, B. (1996). An active testing model for tracking roads in satellite images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18: 1–14.

- Goldman, L., Cook, F., Johnson, P., Brand, D., Rouan, G. and Lee, T. (1996). Prediction of the need for intensive care in patients who come to emergency departments with acute chest pain. *The New England Journal of Medicine*, 334: 1498–504.
- Goldman, L., Weinberg, M., Olshen, R.A., Cook, F., Sargent, R. et al. (1982). A computer protocol to predict myocardial infarction in emergency department patients with chest pain. *The New England Journal of Medicine*, 307: 588–597.
- Golub, T.R., Slonim, D.K., Tamayo, P., Huard, C., Gaasenbeek, M., Mesirov, J.P., Coller, H., Loh, M.L., Downing, J.R., Caligiuri, M.A., Bloomfield, C.D. and Lander, E.S. (1999). Molecular classification of cancer: Class discovery and class prediction by gene expression monitoring. *Science*, 286: 531–537.
- Gordon, L. and Olshen, R.A. (1978). Asymptotically efficient solutions to the classification problem. *Annals of Statistics*, 6: 515–533.
- Gordon, L. and Olshen, R.A. (1980). Consistent nonparametric regression from recursive partitioning schemes. *Journal Multivariate Analysis*, 10: 611–627.
- Gordon, L. and Olshen, R.A. (1984). Almost surely consistent nonparametric regression from recursive partitioning schemes. *Journal Multivariate Analysis*, 15: 147–163.
- Gordon, L. and Olshen, R.A. (1985). Tree-structured survival analysis. *Cancer Treatment Reports*, 69: 1065–1069.
- Huang, X., Chen, S.D. and Soong, S.J. (1998). Piecewise exponential survival trees with time-dependent covariates. *Biometrics*, 54: 1420–14333.
- Inoue, K., Slaton, J.W., Karashima, T., Shuin, T., Sweeney, P., Millikan, R. and Dinney, C.P. (2000). The prognostic value of angiogenesis factor expression for predicting recurrence and metastasis of bladder cancer after neoadjuvant chemotherapy and radical cystectomy. *Clinical Cancer Research*, 6: 4866–4873.
- Intrator, O. and Kooperberg, C. (1995). Trees and splines in survival analysis. *Statistical Methods in Medical Research*, 4: 237–262.
- Kwak, L.W., Halpern, J., Olshen, R.A. and Horning, S.J. (1990). Prognostic significance of actual dose intensity in diffuse large-cell lymphoma: results of a tree-structured survival analysis. *Journal of Clinical Oncology*, 8: 963–977.
- LeBlanc, M. and Crowley, J. (1992). Relative risk trees for censored survival data. *Biometrics*, 48: 411–425.
- LeBlanc, M. and Crowley, J. (1993). Survival trees by goodness-of-split. *Journal of the American Statistical Association*, 88: 457–467.
- LeBlanc, M. and Crowley, J. (1995). A review of tree-based prognostic models. In Thall, P.F. (ed), *Recent Advances in Clinical Trial Design and Analysis*, pp.113–124, Kluwer, New York.
- Levin, N., Zahavi, J. and Olitsky, M. (1995). Amos – A probability-driven, customer-oriented decision support system for target marketing of solo mailings. *European Journal of Operational Research*, 87: 708–721.
- Loh, W.Y. and Vanichsetakul, N. (1988). Tree-structured classification via generalized discriminant analysis. *Journal of the American Statistical Association*, 83: 715–725.

- Long, W.L., Griffith, J.L., Selker, H.P. and D'Agostino, R.B. (1993). A comparison of logistic regression to decision tree induction in a medical domain. *Computers and Biomedical Research*, 26: 74–97.
- Lugosi, G. and Nobel, A.B. (1996). Consistency of data-driven histogram methods for density estimation and classification. *Annals of Statistics*, 24: 687–706.
- Miller, R.G. (1981). *Survival Analysis*, Wiley, New York.
- Mingers, J. (1989). An empirical comparison of selection measures for decision-tree induction. *Machine Learning*, 3: 319–342.
- Mingers, J. (1989). An empirical comparison of pruning methods for decision-tree induction. *Machine Learning*, 4: 227–243.
- Morgan, J.N. and Sonquist, J.A. (1963). Problems in the analysis of survey data and a proposal. *Journal of the American Statistical Association*, 58: 415–434.
- Nagata, K., Okano, Y. and Nozawa, Y. (1997). Differential expression of low Mr GTP-binding proteins in human megakaryoblastic leukemia cell line, MEG-01 and their possible involvement in the differentiation process. *Thrombosis and Haemostasis*, 77: 368–375.
- Nobel, A.B. (1996). Histogram regression estimation using data-dependent partitions. *Annals of Statistics*, 24: 1084–1105.
- Nobel, A.B. and Olshen, R.A. (1996). Termination and continuity of greedy growing for tree structured vector quantizers. *IEEE Transactions on Information Theory*, 42: 191–206.
- Owens, E.A., Griffiths, R.E. and Ratnatunga, K.U. (1996). Using oblique decision trees for the morphological classification of galaxies. *Monthly Notices of the Royal Astronomical Society*, 281: 153–157.
- Pace, R.K. (1995). Parametric, semiparametric and nonparametric estimation of characteristic values within mass assessment and hedonic pricing models. *Journal of Real Estate, Finance and Economics*, 11: 195–217.
- Quinlan, J.R. (1986). Induction of decision trees. *Machine Learning*, 1: 81–106.
- Quinlan, J.R. (1989). Unknown attribute values in induction. In *Proceedings of the Sixth International Machine Learning Workshop*, Morgan Kaufmann, Cornell, New York.
- Segal, M.R. (1988). Regression trees for censored data. *Biometrics*, 44: 35–48.
- Segal, M.R. (1992). Tree-structured methods for longitudinal data. *Journal of American Statistical Association*, 87: 407–418.
- Segal, M.R. (1995). Extending the elements of tree-structured regression. *Statistical Methods in Medical Research*, 4: 219–236.
- Segal, M.R. and Bloch, D.A. (1989). A comparison of estimated proportional hazards models and regression trees. *Statistics in Medicine*, 8: 539–550.
- Selker, H.P., Griffith, J.L., Patil, S., Long, W.L. and D'Agostino, R.B. (1995). A comparison of performance of mathematical predictive methods for medical diagnosis: Identifying acute cardiac ischemia among emergency department patients. *Journal of Investigative Medicine*, 43: 468–476.
- Sitaram, V.S., Huang, C.M. and Israelsen, P.D. (1994). Efficient codebooks for vector quantization image compression with an adaptive tree-search algorithm. *IEEE Transactions on Communications*, 42: 3027–3033.

- Therneau, T.M., Grambsch, P.M. and Fleming, T.R. (1990). Martingale-based residuals for survival models. *Biometrika*, 77: 147–160.
- Toshina, K., Hirata, I., Maemura, K., Sasaki, S., Murano, M., Nitta, M., Yamauchi, H., Nishikawa, T., Hamamoto, N. and Katsu, K. (2000). Enprostil, a prostaglandin-E-2 analogue, inhibits interleukin-8 production of human colonic epithelial cell lines. *Scandinavian Journal of Immunology*, 52: 570–575.
- Wasson, J.H., Sox, H.C., Neff, R.K. and Goldman, L. (1985). Clinical prediction rules: Applications and methodologic standards. *The New England Journal of Medicine*, 313: 793–799.
- Yeates, L.C. and Powis, G. (1997). The expression of the molecular chaperone calnexin is decreased in cancer cells grown as colonies compared to monolayer. *Biochemical and Biophysical Research Communications*, 238: 66–70.
- Zhang, H.P. (1995). Splitting criteria in survival trees. In *Statistical Modelling: Proceedings of the 10th International Workshop on Statistical Modeling*, pp.305–314, Springer.
- Zhang, H.P. (1998). Classification trees for multiple binary responses. *Journal of the American Statistical Association*, 93: 180–193.
- Zhang, H.P. and Bracken, M.B. (1995). Tree-based risk factor analysis of preterm delivery and small-for-gestational-age birth. *American Journal of Epidemiology*, 141: 70–78.
- Zhang, H.P. and Bracken, M.B. (1996). Tree-based, two-stage risk factor analysis for spontaneous abortion. *American Journal of Epidemiology*, 144: 989–996.
- Zhang, H.P., Crowley, J., Sox, H. and Olshen, R.A. (2001). Tree structural statistical methods. *Encyclopedia of Biostatistics*, 6: pp.4561–4573, Wiley, Chichester, England.
- Zhang, H.P., Holford, T. and Bracken, M.B. (1996). A tree-based methods of analysis for prospective studies. *Statistics in Medicine*, 15: 37–49.
- Zhang, H.P. and Singer, B. (1999). *Recursive Partitioning in the Health Sciences*, Springer, New York.
- Zhang, H.P., Yu, C.Y. and Singer, B. (2003). Cell and tumor classification using gene expression data: Construction of forests. *Proceedings of the National Academy of Sciences*, 100: 4168–4172.
- Zhang, H.P., Yu, C.Y., Singer, B. and Xiong, M.M. (2001). Recursive partitioning for tumor classification with gene expression microarray data. *Proceedings of the National Academy of Sciences*, 98: 6730–6735.
- Zhao, L.P. and Prentice, R.L. (1990). Correlated binary regression using a quadratic exponential model. *Biometrika*, 77: 642–648.

Support Vector Machines

III.15

Sebastian Mika, Christin Schäfer, Pavel Laskov, David Tax,
Klaus-Robert Müller

15.1	<i>Introduction</i>	843
15.2	<i>Learning from Examples</i>	843
	General Setting of Statistical Learning	843
	Desirable Properties for Induction Principles.....	844
	Structural Risk Minimization	847
15.3	<i>Linear SVM: Learning Theory in Practice</i>	849
	Linear Separation Planes	849
	Canonical Hyperplanes and Margins	849
15.4	<i>Non-linear SVM</i>	851
	The Kernel Trick.....	852
	Feature Spaces	854
	Properties of Kernels.....	856
15.5	<i>Implementation of SVM</i>	857
	Basic Formulations	857
	Decomposition	861
	Incremental Support Vector Optimization	865
15.6	<i>Extensions of SVM</i>	867
	Regression	867
	One-Class Classification	868

15.7	<i>Applications</i>	870
	Optical Character Recognition (OCR)	870
	Text Categorization and Text Mining	870
	Active Learning in Drug Design	871
	Other Applications	871
15.8	<i>Summary and Outlook</i>	871

Introduction

15.1

In this chapter we introduce basic concepts and ideas of the Support Vector Machines (SVM). In the first section we formulate the learning problem in a statistical framework. A special focus is put on the concept of consistency, which leads to the principle of structural risk minimization (SRM). Application of these ideas to classification problems brings us to the basic, linear formulation of the SVM, described in Sect. 15.3. We then introduce the so called “kernel trick” as a tool for building a non-linear SVM (Sect. 15.4). The practical issues of implementation of the SVM training algorithms and the related optimization problems are the topic of Sect. 15.5. Extensions of the SVM algorithms for the problems of non-linear regression and novelty detection are presented in Sect. 15.6. A brief description of the most successful applications of the SVM is given in Sect. 15.7. Finally, in the last Sect. 15.8 we summarize the main ideas of the chapter.

Learning from Examples

15.2

General Setting of Statistical Learning

15.2.1

The main objective of statistical learning is to find a description of an unknown dependency between measurements of objects and certain properties of these objects. The measurements, to be also called “input variables”, are assumed to be observable in all objects of interest. On the contrary, the objects’ properties, or “output variables”, are in general available only for a small subset of objects known as *examples*. The purpose of estimating the dependency between the input and output variables is to be able to determine the values of output variables for any object of interest.

The problem of estimating an unknown dependency occurs in various practical applications. For example, the input variables can be the prices for a set of stocks and the output variable the direction of change in a certain stock price. As another example, the input can be some medical parameters and the output the probability of a patient having a certain disease. An essential feature of statistical learning is that the information is assumed to be contained in a limited set of examples (the sample), and the estimated dependency should be as accurate as possible for *all* objects of interest.

To proceed with a formal description of main properties of statistical learning, let us fix some notation. Let \mathcal{X} denote the space of input variables representing the objects, and let \mathcal{Y} be the space of output variables. The structure of \mathcal{Y} defines the learning task. For example, if $\mathcal{Y} = \mathbb{R}$, the learning amounts to a regression problem, for $\mathcal{Y} = \{1, 2, 3\}$, the task is a classification problem with three classes, etc.

Let $\mathcal{Z} = \{(x_i, y_i) \in \mathcal{X} \times \mathcal{Y} | i = 1, \dots, M\}$ be a given sample. We assume that there exists some unknown but fixed probability distribution $P(X, Y)$ over the

space $\mathcal{X} \times \mathcal{Y}$ generating our data; that is, $(\mathbf{x}_i, y_i) \in \mathcal{Z}$ are drawn identically and independently from $P(X, Y)$.

The dependency to be estimated takes the form of a function $f : \mathcal{X} \rightarrow \mathcal{Y}$. To decide which of many possible functions best describes the dependency observed in the training sample, we introduce the concept of a *loss function*:

$$\ell : \mathcal{Y} \times \mathcal{Y} \rightarrow \mathbb{R} . \quad (15.1)$$

Such a loss function should be bounded from below and should measure the cost $\ell(f(\mathbf{x}), y)$ of discrepancy between the predicted value $f(\mathbf{x}) \in \mathcal{Y}$ and the true value $y \in \mathcal{Y}$. Then the *risk*, i.e. the expected loss incurred from using a particular prediction function f , can be defined as:

$$R(f) = \mathbb{E}_P[\ell(f(\mathbf{x}), y)] , \quad (15.2)$$

where \mathbb{E}_P denotes the expectation with respect to the joint distribution $P(X, Y)$ of input and output variables.

Notice that, if we would know the joint distribution $P(X, Y)$, the learning problem can be easily solved. For example, in the classification case one could calculate the conditional probability $P(Y|X)$ and compute the so called “Bayes-optimal solution”:

$$f^*(\mathbf{x}) = \operatorname{argmax}_{y_1 \in \mathcal{Y}} \int_{y_2 \in \mathcal{Y}} \ell(y_1, y_2) P(Y = y_2 | X = \mathbf{x}) . \quad (15.3)$$

However, in our setup $P(X, Y)$ is unknown, and only a sample \mathcal{Z} is available. One possible solution would be to estimate $P(X, Y)$ or $P(Y|X)$ from the sample \mathcal{Z} . In many theoretical and practical approaches the inference is carried out exactly in this way (Duda et al., 2001; Bishop, 1995; Devroye et al., 1996). But it is also well known that estimating a density from empirical data is a hard problem, especially in the multi-dimensional case. The number of examples one needs in order to get a reliable estimate of a density in N dimensions grows exponentially with N . In the approach to be followed in this chapter we shall attempt to estimate the function f directly from \mathcal{Z} without using $P(X, Y)$ or $P(Y|X)$. For this, the following three steps are necessary. First, a class of functions \mathcal{F} needs to be defined. Second, a suitable loss ℓ is to be fixed. Finally, a method has to be provided to find the function f which minimizes the risk $R(f)$ among all $f \in \mathcal{F}$. Such method is called an “induction principle”. Desirable properties of such an induction principle are discussed in the next section.

15.2.2 Desirable Properties for Induction Principles

The most commonly used induction principle is the one of minimizing the empirical risk

$$R_{\text{emp}}(f) = \frac{1}{M} \sum_{i=1}^M \ell(f(\mathbf{x}_i), y_i) , \quad (15.4)$$

which is the empirical counterpart of the expected risk (15.2). The goal of learning in our setup is to find an algorithm that, given a training sample \mathcal{Z} , finds a function $f \in \mathcal{F}$ that minimizes (15.4). Notice that this will not necessarily result in a unique solution. As one can see in Fig. 15.1 more than one function can have the same (e.g. zero) empirical risk on the same data sample. However, these functions can take arbitrary values at other points in \mathcal{X} ; hence the solution that minimizes the empirical risk is not guaranteed to minimize the true risk (15.2).

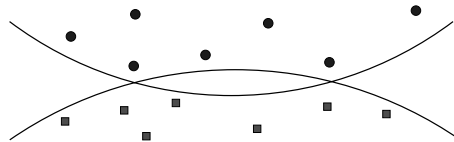


Figure 15.1. Two functions that separate two classes of data points with zero empirical risk. Without further information it is impossible to decide for one of them

The other two phenomena arising in relation with the minimization of the empirical risk (15.4) are over- and under-fitting. An overly complex function f might describe the training data well but does not generalize to unseen examples. The converse could also happen. Assume the function class \mathcal{F} we can choose from is very small, e.g. it contains only a single, fixed function. Then our learning machine would trivially be consistent, since $R(f) = \text{const}$ for all $f \in \mathcal{F}$. But if this single $f \in \mathcal{F}$ is not by accident the rule that generates our data, the decisions are unrelated to the concept generating our data. This phenomenon is called under-fitting (cf. Fig. 15.2). Apparently we need some way of controlling how large the class of functions \mathcal{F} is, such that we avoid over- and under-fitting and obtain solutions that generalize well (i.e. with reasonable complexity). The questions of consistency, over- and under-fitting are closely related and will lead us to a concept known as regularization (e.g. Tikhonov and Arsenin, 1977; Morozov, 1984) and to the principle of structural risk minimization (Vapnik, 1998).

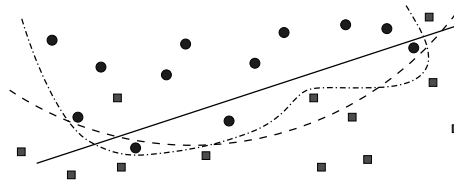


Figure 15.2. An illustration of under- and over-fitting on a small sample. The simple linear function (solid line) underfits the data and already makes training errors. The complex one (dash-dotted line) has no training error but may not generalize well on unseen data. The function with intermediate complexity (dashed line) seems to capture the decision boundary best

Regularization

In the previous paragraphs we have shown that for successful learning it is not enough to find a function with minimal empirical risk. If we are interested in a good estimation of the true risk on all possible datapoints, we need to introduce a complexity control and choose our solution by minimizing the following objective function:

$$R_{\text{emp}}(f, \mathcal{Z}) + \lambda \Omega(f) . \quad (15.5)$$

This equation shows a regularization approach. We add a penalty term to make the trade-off between the complexity of the function class and the empirical error. Using such a regularization a bound for the true risk can be derived.

There are several possibilities to choose λ and Ω in order to derive a consistent inductive principle. In the following sections we will describe the choice inspired by the work of Vapnik. Other possible choices are for example Akaike information criterion (Akaike, 1974) or Mallows Cp (Mallows, 1973), used in classical statistics, as well as spline-regularization (Wahba, 1980), wavelet regularization (Donoho et al., 1996), CART (Breiman et al., 1984) and many other modern approaches. A general foundation for regularization in model selection is given in (Barron et al., 1999). Bartlett and Mendelson (2002) investigate regularization in the context of SVM.

Consistency

Let us define more closely what consistency means and how it can be characterized. Let us denote by f^M the function $f \in \mathcal{F}$ that minimizes (15.4) for a given training sample \mathcal{Z} of size M . The notion of *consistency* implies that, as $M \rightarrow \infty$, $|R(f^M) - R_{\text{emp}}(f^M)| \rightarrow 0$ in probability. We have already seen in a previous example that such convergence may not be the case in general, the reason being that f^M now depends on the sample \mathcal{Z} . One can show that a necessary and sufficient condition for consistency is *uniform* convergence, over all functions in \mathcal{F} , of the difference between the expected and the empirical risk to zero. This insight is summarized in the following theorem:

1

Theorem 1: (Vapnik and Chervonenkis, 1991)

One-sided uniform convergence in probability, i.e.

$$\lim_{M \rightarrow \infty} \mathbb{P} \left[\sup_{f \in \mathcal{F}} (R(f) - R_{\text{emp}}(f)) > \varepsilon \right] = 0 , \quad (15.6)$$

for all $\varepsilon > 0$, is a necessary and sufficient condition for (nontrivial) consistency of empirical risk minimization.

Since the condition in the theorem is not only sufficient but also necessary it seems reasonable that any “good” learning machine implementing a specific function class should satisfy condition (15.6).

Structural Risk Minimization

15.2.3

Consequently, the question arises how one can choose function classes that satisfy Theorem 1 in practice? It will turn out that this is possible and it crucially depends on the question how complex the functions in the class \mathcal{F} are, a question we have already seen to be equally important when talking about over- and under-fitting. But what does complexity mean and how can one *control* the size of a function class?

The complexity of a function class can be measured by the number of different possible combinations of outcome assignments when choosing functions from this class. This quantity is usually difficult to obtain theoretically for useful classes of functions. Popular approximations of this measure are covering numbers (Shawe-Taylor et al., 1998), annealed entropy and fat-shattering dimension (Bartlett et al., 1996), VC-entropy and VC dimension (Vapnik, 1998), or Rademacher and Gaussian complexity (Bartlett and Mendelson, 2002). We will not go into detail about these quantities here.

A specific way of controlling the complexity of a function class is given by the VC-theory and the structural risk minimization (SRM) principle (Vapnik, 1998). Here the concept of complexity is captured by the VC-dimension h of the function class \mathcal{F} . Roughly speaking, the VC-dimension measures how many (training) points can be shattered (i.e. separated for all possible labellings) using functions of the class. This quantity can be used to bound the probability that the expected error deviates much from the empirical error for any function from the class, i.e. VC-style bounds usually take the form

$$\left[\sup_{f \in \mathcal{F}} (R(f) - R_{\text{emp}}(f, \mathcal{Z})) > \varepsilon \right] \leq H(\mathcal{F}, M, \varepsilon), \quad (15.7)$$

where H is some function that depends on properties of the function class \mathcal{F} , e.g. the VC-dimension, the size M of the training set and the desired closeness ε . By equating the right-hand side of (15.7) to $\delta > 0$ and solving $H = \delta$ for ε one can turn these bounds into expressions of the following form: with probability at least $1 - \delta$ over the random draw of the training sample \mathcal{Z} ,

$$R(f) \leq R_{\text{emp}}(f, \mathcal{Z}) + \tilde{H}(\mathcal{F}, M, \delta), \quad (15.8)$$

where \tilde{H} is the penalty term that measures our degree of uncertainty. If the function class is simple then \tilde{H} is small. This penalty term usually increases if we require a higher precision (e.g. with $\log(1/\delta)$) and decreases if we observe more examples (e.g. with $1/M$ or $1/\sqrt{M}$). Note that this prototypical bound is structurally identical to the regularized risk functional in (15.5). The practical implication of bounds like (15.8) is that our learning machine should be constructed such that

1. it finds a function with a small empirical error, and
2. at the same time keeps the penalty term \tilde{H} small.

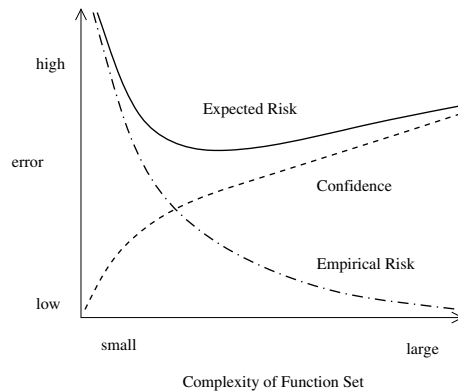


Figure 15.3. Schematic illustration of (15.8). The *dash-dotted line* represents the training error (empirical risk), the *dashed line* the upper bound on the complexity term (confidence). With higher complexity the empirical error decreases but the upper bound on the risk uncertainty becomes worse. For a certain complexity of the function class the best expected risk (*solid line*) is obtained. Thus, in practice the goal is to find the best trade-off between empirical error and complexity

Only if our learning principle can control both quantities we have a guarantee that the expected error of our estimate will be small (cf. Fig. 15.3).

One of the most famous learning bounds is due to Vapnik and Chervonenkis:

2

Theorem 2: (Vapnik and Chervonenkis, 1974)

Let h denote the VC-dimension of the function class \mathcal{F} and let R_{emp} be defined by (15.4) using the 0/1-loss. For all $\delta > 0$ and $f \in \mathcal{F}$ the inequality bounding the risk

$$R(f) \leq R_{\text{emp}}(f, \mathcal{Z}) + \sqrt{\frac{h \left(\ln \frac{2M}{h} + 1 \right) - \ln(\delta/4)}{M}} \quad (15.9)$$

holds with probability of at least $1 - \delta$ for $M > h$ over the random draw of the training sample \mathcal{Z} .

This theorem lays the ground for the Support Vector algorithm that we will consider in more detail in Sect. 15.3.

As a consequence of Theorem 2 the so called “Structural Risk Minimization principle” (SRM) is suggested (i.e. Cortes and Vapnik, 1995; Vapnik, 1998). According to this principle a nested family of function classes $\mathcal{F}_1 \subseteq \dots \subseteq \mathcal{F}_k$ with non-decreasing VC-dimension $h_1 \leq \dots \leq h_k$, is constructed. After the solutions f_1, \dots, f_k of the empirical risk minimization (15.4) in the function classes $\mathcal{F}_1, \dots, \mathcal{F}_k$ have been found, the SRM principle chooses the function class \mathcal{F}_i (and the function f_i) such that an upper bound on the generalization error like (15.9) is minimized.

Linear SVM: Learning Theory in Practice

15.3

Having summarized the prerequisites from statistical learning theory, we now give an example of a particular learning machine that builds upon these insights. The Support Vector Machine algorithm (SVM) developed by Vapnik and others (e.g. Boser et al., 1992; Cortes and Vapnik, 1995; Vapnik, 1998; Cristianini and Shawe-Taylor, 2000; Müller et al., 2001; Schölkopf and Smola, 2002, and numerous others) is one of the most successful classification techniques over the last decade, especially after being combined with the kernel idea which we shall discuss in Sect. 15.4.

Linear Separation Planes

15.3.1

We are now going to discuss how one could possibly control the size of a function class and how to select the empirical risk minimizer in this class.

In the following, let us assume that we are dealing with a two class classification problem (i.e. $\mathcal{Y} = \{-1, +1\}$) in a real-valued vector space, e.g. $\mathcal{X} = \mathbb{R}^N$. Further, we assume that the distribution of these two classes is such that they are linearly separable, i.e. one can find a linear function of the inputs $\mathbf{x} \in \mathcal{X}$ such that $f(\mathbf{x}) < 0$ whenever the label $y = -1$ and $f(\mathbf{x}) \geq 0$ otherwise. This can be conveniently expressed by a hyperplane in the space \mathcal{X} , i.e. we are looking for a function f of the form

$$f(\mathbf{x}) = (\mathbf{w}^\top \mathbf{x}) + b. \quad (15.10)$$

Assume that the function class \mathcal{F} we choose our solution from is the one containing all possible hyperplanes, i.e. $\mathcal{F} = \{f : \mathcal{X} \rightarrow \mathbb{R} \mid f(\mathbf{x}) = (\mathbf{w}^\top \mathbf{x}) + b\}$. For $\mathcal{X} = \mathbb{R}^N$ it is rather straightforward to show that the VC-dimension of this class of functions will be $h = N + 1$, i.e. in an N dimensional space the maximal number of points that can be separated for an arbitrary labelling using a hyperplane is $N + 1$.

Canonical Hyperplanes and Margins

15.3.2

To apply the SRM principle in practice the VC-dimension of the class of hyperplanes must be finite, and a nested structure of function classes must be defined. To this end we define the function classes

$$\mathcal{F}_\Lambda = \{f : \mathbb{R}^N \rightarrow \mathbb{R} \mid f(\mathbf{x}) = (\mathbf{w}^\top \mathbf{x}) + b, \|\mathbf{w}\| \leq \Lambda\}. \quad (15.11)$$

Clearly $\mathcal{F}_{\Lambda_1} \subseteq \mathcal{F}_{\Lambda_2}$ whenever $\Lambda_1 \leq \Lambda_2$. But what effect does constraining the norm of the weight vector have on the corresponding VC-dimensions of \mathcal{F}_Λ ? It turns out that we also get $h(\mathcal{F}_{\Lambda_1}) \leq h(\mathcal{F}_{\Lambda_2})$ for $\Lambda_1 \leq \Lambda_2$ (see (15.12) below), i.e. the required structure is present.

The crucial ingredient in making the function classes \mathcal{F}_A nested is to define a unique representation for each hyperplane. We introduce the concept of canonical hyperplanes and the notion of margins. If the data are separable by (w, b) then they are also separable by any (positive) multiple of (w, b) and hence there exist an infinite number of representations for the same separating hyperplane. In particular, all function classes \mathcal{F}_A would have the same VC-dimension as they would contain the same functions in different representations.

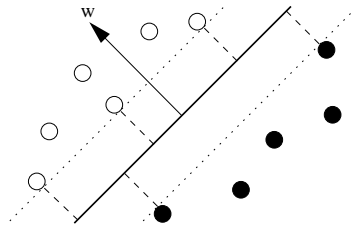


Figure 15.4. Linear classifier and margins. A linear classifier is defined by the normal vector w of a hyperplane and an offset b , i.e. the decision boundary is $\{x | (w^\top x) + b = 0\}$ (solid line). Each of the two half spaces induced by this hyperplane corresponds to one class, i.e. $f(x) = \text{sgn}((w^\top x) + b)$. The margin of a linear classifier is the minimal distance of any training point to the hyperplane. For the case shown in the picture it is the distance between the dotted lines and the solid line

A canonical hyperplane with respect to an M -sample \mathcal{Z} is defined as a function

$$f(x) = (w^\top x) + b,$$

where w is normalized such that

$$\min_{i=1, \dots, M} |f(x_i)| = 1.$$

The notion of a canonical hyperplane is illustrated in Fig. 15.4. Notice that none of the training examples produces an absolute output that is smaller than one and the examples closest the hyperplane have exactly an output of one, i.e. $(w^\top x) + b = \pm 1$. In Sect. 15.5, we will see that the latter objects will be used in the description of the hyperplane, and they are therefore called the *support vectors*. In Fig. 15.4 these are the objects which are connected to the decision boundary (dashed lines). Since we assumed the sample \mathcal{Z} to be linearly separable, we can turn any f that separates the data into a canonical hyperplane by suitably normalizing the weight vector w and adjusting the threshold b correspondingly.

The *margin* is defined to be the minimal Euclidean distance between any training example x_i and the separating hyperplane. Intuitively, the margin measures how good the separation between the two classes by a hyperplane is. If this hyperplane is in the canonical form, the margin can be measured by the length of the weight vector w . Consider two support vectors x_1 and x_2 from different classes. The margin is given by the projection of the distance between these two points on the direction

perpendicular to the hyperplane. This distance can be computed as (e.g. Vapnik, 1998)

$$\left(\frac{\mathbf{w}^\top (\mathbf{x}_1 - \mathbf{x}_2)}{\|\mathbf{w}\|} \right) = \frac{2}{\|\mathbf{w}\|} .$$

The smaller the norm of the weight vector \mathbf{w} in the canonical representation, the larger the margin.

More generally, it was shown (e.g. Vapnik, 1998) that if the hyperplane is constructed under the constraint $\|\mathbf{w}\|_2 \leq \Lambda$ then the VC-dimension of the class \mathcal{F}_Λ is bounded by

$$h \leq \min (\Lambda^2 R^2 + 1, N + 1) , \quad (15.12)$$

where R is the radius of the smallest sphere around the data. Thus, if we bound the margin of a function class from below, say by $2/\Lambda$, we can control its VC-dimension and hence apply the SRM principle as shown in Fig. 15.5.

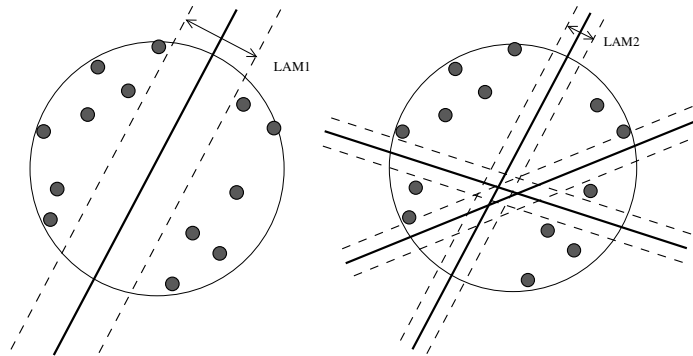


Figure 15.5. Illustration of why a large margin reduces the complexity of a linear hyperplane classifier. If we choose hyperplanes with a large margin, there is only a small number of possibilities to separate the data, i.e. the VC-dimension of \mathcal{F}_{Λ_1} is small (*left panel*). On the contrary, if we allow smaller margins there are more separating hyperplanes, i.e. the VC-dimension of \mathcal{F}_{Λ_2} is large (*right panel*)

A particularly important insight is that the complexity only indirectly depends on the dimensionality of the data. This is very much in contrast to e.g. density estimation, where the problems become more difficult as the dimensionality of the data increases. For SVM classification, if we can achieve a large margin the problem remains simple.

Non-linear SVM

In the previous section we have seen that by restricting ourselves to linear functions one can control the complexity of a learning machine. We have thus avoided the

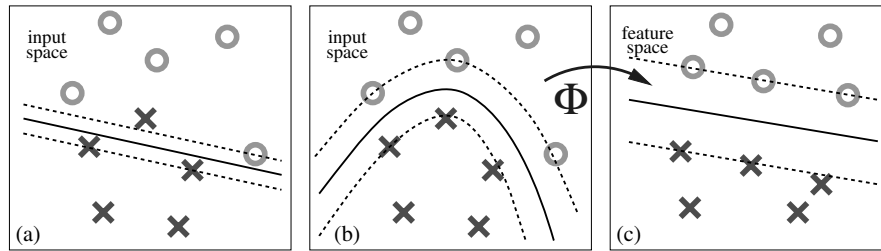


Figure 15.6. Three different views on the same two class separation problem. (a) A linear separation of the input points is not possible without errors. Even allowing misclassification of one data point results in a small margin. (b) A better separation is provided by a non-linear surface in the input space. (c) This non-linear surface corresponds to a linear surface in a feature space. Data points are mapped from input space to feature space by the function Φ induced by the kernel function k

problem of dealing with too complex functions at the price of being able to solve only linearly separable problems. In the following we will show how to extend the linear SVM for constructing a very rich set of non-linear decision functions while at the same time controlling their complexity.

Central to the success of support vector machines was the re-discovery of the so called *Reproducing Kernel Hilbert Spaces* (RKHS) and *Mercer's Theorem* (Boser et al., 1992). There is a large body of literature dealing with kernel functions, their theory and applicability, see e.g. Kolmogorov (1941), Aronszajn (1950), Aizerman et al. (1964), Boser et al. (1992) or Schölkopf and Smola (2002) for an overview. We only recall the basic definitions and properties necessary for turning our linear, hyperplane based learning technique into a very powerful algorithm capable of finding non-linear decision functions with controllable complexity.

15.4.1 The Kernel Trick

The basic idea of the so called *kernel-methods* is to first preprocess the data by some non-linear mapping Φ and then to apply the same linear algorithm as before but in the image space of Φ . (cf. Fig. 15.6 for an illustration). More formally we apply the mapping

$$\begin{aligned}\Phi : \mathbb{R}^N &\rightarrow \mathcal{E}, \\ \mathbf{x} &\mapsto \Phi(\mathbf{x})\end{aligned}$$

to the data $\mathbf{x}_1, \dots, \mathbf{x}_M \in \mathcal{X}$ and consider our algorithm in \mathcal{E} instead of \mathcal{X} , i.e. the sample is preprocessed as

$$\{(\Phi(\mathbf{x}_1), y_1), \dots, (\Phi(\mathbf{x}_M), y_M)\} \subseteq (\mathcal{E} \times \mathcal{Y})^M.$$

In certain applications we might have sufficient knowledge about our problem such that we can design an appropriate Φ by hand (e.g. Zien et al., 2000;

Blankertz et al., 2002). If this mapping is not too complex to compute and the space \mathcal{E} is not too high-dimensional, we might just explicitly apply this mapping to our data. Something similar is done for (single hidden layer) neural networks (Bishop, 1995), radial basis networks (e.g. Moody and Darken, 1989) or Boosting algorithms (Freund and Schapire, 1997), where the input data are mapped to some representation given by the hidden layer, the RBF bumps or the hypotheses space, respectively (Rätsch et al., 2002). The difference with kernel-methods is that for a suitably chosen Φ we get an algorithm that has powerful non-linearities but is still very intuitive and retains most of the favorable properties of its linear input space version.

The problem with explicitly using the mapping Φ to construct a feature space is that the resulting space can be extremely high-dimensional. As an example consider the case when the input space \mathcal{X} consists of images of 16×16 pixels, i.e. 256 dimensional vectors, and we choose 5th order monomials as non-linear features. The dimensionality of such space would be

$$\binom{5 + 256 - 1}{5} \approx 10^{10} .$$

Such a mapping would clearly be intractable to carry out explicitly. We are not only facing the technical problem of storing the data and doing the computations, but we are also introducing problems due to the fact that we are now working in an extremely sparse sampled space. By the use of the SRM principle, we can be less sensitive to the dimensionality of the space and achieve good generalization.

The problems concerning the storage and the manipulation of the high dimensional data can be avoided. It turns out that for a certain class of mappings we are well able to compute scalar products in this new space even if it is extremely high dimensional. Simplifying the above example of computing all 5th order products of 256 pixels to that of computing all 2nd order products of two “pixels”, i.e.

$$\mathbf{x} = (x_1, x_2) \quad \text{and} \quad \Phi(\mathbf{x}) = \left(x_1^2, \sqrt{2}x_1x_2, x_2^2 \right) ,$$

the computation of a scalar product between two such feature space vectors can be readily reformulated in terms of a kernel function k :

$$\begin{aligned} (\Phi(\mathbf{x})^\top \Phi(\mathbf{z})) &= \left(x_1^2, \sqrt{2}x_1x_2, x_2^2 \right) \left(z_1^2, \sqrt{2}z_1z_2, z_2^2 \right)^\top \\ &= ((x_1, x_2)(z_1, z_2)^\top)^2 \\ &= (\mathbf{x}^\top \mathbf{z})^2 \\ &=: k(\mathbf{x}, \mathbf{z}) . \end{aligned}$$

This finding generalizes: For $\mathbf{x}, \mathbf{z} \in \mathbb{R}^N$, and $d \in \mathbb{N}$ the kernel function

$$k(\mathbf{x}, \mathbf{z}) = (\mathbf{x}^\top \mathbf{z})^d$$

computes a scalar product in the space of all products of d vector entries (monomials) of \mathbf{x} and \mathbf{z} (Vapnik, 1998; Schölkopf et al., 1998b).

The *kernel trick* (Aizerman et al., 1964; Boser et al., 1992; Vapnik, 1998) is to take the original algorithm and formulate it such, that we only use $\Phi(\mathbf{x})$ in scalar products. Then, if we can efficiently evaluate these scalar products, we do not need to carry out the mapping Φ explicitly and can still solve the problem in the huge feature space \mathcal{E} . Furthermore, we do not need to know the mapping Φ but only the kernel function.

Now we can ask two questions:

1. For which mappings Φ does there exist a simple way to evaluate the scalar product?
2. Under which conditions does a function $k : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$ correspond to a scalar product?

The first question is difficult to answer in general. But for the second question there exists an answer which we present in the following.

15.4.2 Feature Spaces

To address the question whether a kernel function $k : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$ is a dot-product let us first introduce some more notation and definitions. Given a training sample $\{\mathbf{x}_1, \dots, \mathbf{x}_M\} \subseteq \mathcal{X}$, the $M \times M$ matrix K with elements $K_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$ is called the kernel matrix or the Gram matrix. An $M \times M$ matrix K (and any other symmetric matrix) is said to be positive definite if any quadratic form over K is positive, i.e. for all $r_i \in \mathbb{R}, i = 1, \dots, M$, we have

$$\sum_{i,j=1}^M r_i r_j K_{ij} \geq 0. \quad (15.13)$$

Positive definite kernels are exactly those giving rise to a positive definite kernel matrix k for all M and all sets $\{\mathbf{x}_1, \dots, \mathbf{x}_M\} \subseteq \mathcal{X}$. Note, that for a kernel (and a matrix) to be positive definite, it is necessary to be symmetric and non-negative on the diagonal.

For any positive definite kernel k we can construct a mapping Φ into a feature space \mathcal{E} , such that k acts as a dot-product over Φ . As a matter of fact, it is possible to construct more than one of these spaces. We will omit many crucial details and only present the central results. For more details see e.g. Schölkopf and Smola, 2002.

The Feature Map

Given a real-valued, positive definite kernel function k , defined over a non-empty set \mathcal{X} , we define the feature space \mathcal{E} as the space of all functions mapping from

\mathcal{X} to \mathbb{R} , i.e. as $\mathcal{E} = \mathbb{R}^{\mathcal{X}} = \{f|f : \mathcal{X} \rightarrow \mathbb{R}\}$. Notice that, unlike the example in Fig. 15.6, this feature space is not a usual Euclidean space but rather a vector space of functions. The mapping Φ is now defined as

$$\Phi : \mathcal{X} \rightarrow \mathbb{R}^{\mathcal{X}}, \Phi(\mathbf{x}) = k(\cdot, \mathbf{x}), \quad (15.14)$$

i.e. Φ maps each \mathbf{x} to the function $k(\cdot, \mathbf{x})$, i.e. the kernel k where the first argument is free and the second is fixed to \mathbf{x} . One can show that the set of all linear combinations of the form

$$f(\cdot) = \sum_{i=1}^M \alpha_i k(\cdot, \mathbf{x}_i), \quad (15.15)$$

for arbitrary M , $\alpha_i \in \mathbb{R}$, and $\mathbf{x}_1, \dots, \mathbf{x}_M$ forms a vector space. Especially, for all functions of the form (15.15) one gets

$$\langle k(\cdot, \mathbf{x}), f \rangle_{\mathcal{H}} = f(\mathbf{x}),$$

where $\langle \cdot, \cdot \rangle_{\mathcal{H}}$ denotes the scalar product in some Hilbert space that will become clearer below. In particular we have

$$\begin{aligned} \langle k(\cdot, \mathbf{x}), k(\cdot, \mathbf{z}) \rangle_{\mathcal{H}} &= \langle \Phi(\mathbf{x}), \Phi(\mathbf{z}) \rangle_{\mathcal{E}} \\ &= k(\mathbf{x}, \mathbf{z}). \end{aligned}$$

The last property is the reason why positive definite kernels are also called reproducing kernels: they reproduce the evaluation of f on \mathbf{x} . It also shows that k indeed computes, as desired, the dot-product in \mathcal{E} for $\Phi(\mathbf{x})$ defined as in (15.14). Hence (15.14) is one possible realization of the mapping associated with a kernel and is called the feature map (for its empirical counterpart see e.g. (Mika, 2002)). The following is a formal definition of a *Reproducing Kernel Hilbert Space* (cf. Schölkopf and Smola, 2002).

Definition 1: *Reproducing Kernel Hilbert Space (RKHS)*

Let \mathcal{X} be a nonempty set and \mathcal{H} a Hilbert space of functions $f : \mathcal{X} \rightarrow \mathbb{R}$. Then \mathcal{H} is called a *reproducing kernel Hilbert space* endowed with the dot product $\langle \cdot, \cdot \rangle$ if there exists a function $k : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$ with the properties that

1. k has the reproducing property $\langle f, k(\cdot, \mathbf{x}) \rangle = f(\mathbf{x})$ for all $f \in \mathcal{H}$, in particular $\langle k(\cdot, \mathbf{x}), k(\cdot, \mathbf{z}) \rangle = k(\mathbf{x}, \mathbf{z})$, and
 2. k spans \mathcal{H} , i.e. $\mathcal{H} = \overline{\text{span}\{k(\cdot, \mathbf{x}) | \mathbf{x} \in \mathcal{X}\}}$, where \bar{A} denotes the completion of the set A .
-

One can show, that the kernel k for such a RKHS is uniquely determined.

Mercer Kernels

As a second way to identify a feature space associated with a kernel k one can use a technique derived from Mercer's Theorem.

The Mercer's Theorem, which we will reproduce in the following, states that if the function k (the kernel) gives rise to a positive integral operator, the evaluation

of $k(\mathbf{x}, \mathbf{z})$ can be expressed as a finite or infinite, absolute and uniformly convergent series, almost everywhere. This series then defines in another way a feature space and an associated mapping connected to the kernel k .

Let \mathcal{X} be a finite measure space, i.e. a space with a σ -algebra and a measure μ satisfying $\mu(\mathcal{X}) \leq \infty$.

3

Theorem 3: (Mercer, 1909)

Suppose $k \in L_\infty(\mathcal{X}^2, \mu)$ is a symmetric real-valued function such that the integral operator

$$T_k : L_2(\mathcal{X}, \mu) \rightarrow L_2(\mathcal{X}, \mu), \quad (T_k f)(\mathbf{x}) := \int_{\mathcal{X}} k(\mathbf{x}, \mathbf{z}) f(\mathbf{z}) d\mu(\mathbf{z})$$

is positive definite, i.e. for all $f \in L_2(\mathcal{X}, \mu)$

$$\int_{\mathcal{X}^2} k(\mathbf{x}, \mathbf{z}) f(\mathbf{x}) f(\mathbf{z}) d\mu(\mathbf{x}) d\mu(\mathbf{z}) \geq 0.$$

Let $\varphi_j \in L_2(\mathcal{X}, \mu)$ be the normalized orthogonal eigenfunctions of T_k associated with the eigenvalues $\lambda_j \geq 0$, sorted in non-increasing order. Then

1. $(\lambda_j)_j \in l_1$
2. $k(\mathbf{x}, \mathbf{z}) = \sum_{j=1}^{N_\varepsilon} \lambda_j \varphi_j(\mathbf{x}) \varphi_j(\mathbf{z})$ holds for almost all \mathbf{x}, \mathbf{z} . Either $N_\varepsilon \in \mathbb{N}$ or $N_\varepsilon = \infty$; in the latter case, the series converges absolutely and uniformly for almost all \mathbf{x}, \mathbf{z} .

If we choose as feature space $\mathcal{E} = l_2^{N_\varepsilon}$ and the mapping Φ as

$$\Phi : \mathcal{X} \rightarrow l_2^{N_\varepsilon}, \quad \Phi(\mathbf{x}) = \left(\sqrt{\lambda_j} \varphi_j(\mathbf{x}) \right)_{j=1, \dots, N_\varepsilon},$$

we see from the second statement in Theorem 3 that the kernel k corresponds to the dot product in $l_2^{N_\varepsilon}$, i.e. $k(\mathbf{x}, \mathbf{z}) = \langle \Phi(\mathbf{x}), \Phi(\mathbf{z}) \rangle$.

The kernels satisfying the Mercer's Theorem are called *Mercer kernels*. It can be shown that, if the set \mathcal{X} on which the kernel is defined, is compact, a kernel is a Mercer kernel if and only if it is a positive definite kernel (cf. Smola et al., 1998).

Table 15.1 lists some of the most widely used kernel functions. More sophisticated kernels (e.g. kernels generating splines or Fourier expansions) and kernels designed for special applications like DNA analysis can be found in Vapnik (1998), Stitson et al. (1997), Smola et al. (1998), Haussler (1999), Jaakkola et al. (2000), Zien et al. (2000), Tsuda et al. (2002) and numerous others.

15.4.3 Properties of Kernels

Besides being useful tools for the computation of dot-products in high- or infinite-dimensional spaces, kernels possess some additional properties that make them an interesting choice in algorithms. It was shown (Girosi et al., 1993) that using a particular positive definite kernel corresponds to an *implicit* choice of a regularization

Table 15.1. Common kernel functions

Gaussian RBF	$k(\mathbf{x}, \mathbf{z}) = \exp\left(\frac{-\ \mathbf{x} - \mathbf{z}\ ^2}{c}\right)$	(15.16)
Polynomial	$k(\mathbf{x}, \mathbf{z}) = ((\mathbf{x}^\top \mathbf{z}) + \theta)^d$	(15.17)
Sigmoidal	$k(\mathbf{x}, \mathbf{z}) = \tanh(\kappa(\mathbf{x}^\top \mathbf{z}) + \theta)$	
Inverse multi-quadric	$k(\mathbf{x}, \mathbf{z}) = \frac{1}{\sqrt{\ \mathbf{x} - \mathbf{z}\ ^2 + c^2}}$	

operator. For translation-invariant kernels, the regularization properties can be expressed conveniently in Fourier space in terms of the frequencies (Smola et al., 1998; Girosi, 1998). For example, Gaussian kernels (cf. (15.16)) correspond to a general smoothness assumption in all k -th order derivatives (Smola et al., 1998). Vice versa, using this correspondence, kernels matching a certain prior about the frequency content of the data can be constructed so as to reflect our prior problem knowledge.

Another particularly useful feature of kernel functions is that we are not restricted to kernels that operate on vectorial data, e.g. $\mathcal{X} = \mathbb{R}^N$. In principle, it is also possible to define positive kernels for e.g. strings or graphs, i.e. making it possible to embed discrete objects into a metric space and apply metric-based algorithms (e.g. Haussler, 1999; Watkins, 2000; Zien et al., 2000).

Furthermore, many algorithms can be formulated using so called *conditionally positive definite* kernels (cf. Smola et al., 1998; Schölkopf, 2001) which are a superclass of the positive definite kernels considered so far. They can be interpreted as generalized non-linear dissimilarity measures (opposed to just the scalar product) and are applicable e.g. in SVM and kernel PCA.

Implementation of SVM

15.5

Basic Formulations

15.5.1

We are now at the point to merge the ideas of statistical learning, Structural Risk Minimization and reproducing kernels into a single algorithm, Support Vector

Machines, suitable for a wide range of practical application. The main goal of this algorithm is to find a weight vector \mathbf{w} separating the data \mathcal{Z} with the largest possible margin.

Separable Data

Assume that the data are separable. Our goal is to find the smallest possible \mathbf{w} without committing any error. This can be expressed by the following quadratic optimization problem:

$$\begin{aligned} \min_{\mathbf{w}, b} \quad & \frac{1}{2} \|\mathbf{w}\|^2, \\ \text{subject to} \quad & y_i ((\mathbf{w}^\top \mathbf{x}_i) + b) \geq 1, \quad \forall i = 1, \dots, M. \end{aligned} \quad (15.18)$$

The constraints in (15.18) assure that \mathbf{w} and b are chosen such that no example has a distance to the hyperplane smaller than one. The problem can be solved directly by using a quadratic optimizer. Notice that the optimal solution renders a canonical hyperplane. In contrast to many neural networks (e.g. Bishop, 1995) one can always find the *global* minimum. In fact, all minima of (15.18) are global minima, although they might not be unique as e.g. in the case when $M < N$, where N is the dimensionality of the data.

In the formulation (15.18), referred to as the primal formulation, we are bound to use the original data \mathbf{x} . In order to apply the kernel trick (cf. Sect. 15.4.1) we need to transform the problem such that the only operation involving the original data \mathbf{x} is an inner product between certain data points. This can be achieved by transforming the problem to the dual formulation. The notion of duality is an essential part of non-linear optimization theory, for details one can refer to any standard textbook on mathematical programming (e.g. Luenberger, 1973; Bertsekas, 1995). For our purposes it is important that for every quadratic optimization problem there exists a dual problem which is also a quadratic problem. If both the primal and the dual problems have an optimal solution then the values of the objective function at the optimal solutions coincide. This implies that by solving the dual problem – which uses the original data \mathbf{x} only through inner products – the solution to the primal problem can be reconstructed.

To derive the dual of (15.18), we introduce Lagrange multipliers $\alpha_i \geq 0$ with $i = 1, \dots, M$, one for each of the constraints in (15.18). We obtain the following Lagrangian:

$$L(\mathbf{w}, b, \boldsymbol{\alpha}) = \frac{1}{2} \|\mathbf{w}\|^2 - \sum_{i=1}^M \alpha_i (y_i ((\mathbf{w}^\top \mathbf{x}_i) + b) - 1). \quad (15.19)$$

The task is to minimize (15.19) with respect to \mathbf{w} , b and to maximize it with respect to α_i . At the optimal point, we have the following saddle point equations:

$$\frac{\partial L}{\partial b} = 0 \quad \text{and} \quad \frac{\partial L}{\partial \mathbf{w}} = 0,$$

which translate into

$$\sum_{i=1}^M \alpha_i y_i = 0 \quad \text{and} \quad \mathbf{w} = \sum_{i=1}^M \alpha_i y_i \mathbf{x}_i . \quad (15.20)$$

From the right equation of (15.20), we find that \mathbf{w} is contained in the subspace spanned by the \mathbf{x}_i in the training set. By substituting (15.20) into (15.19), we get the dual quadratic optimization problem:

$$\max_{\alpha} \sum_{i=1}^M \alpha_i - \frac{1}{2} \sum_{i,j=1}^M \alpha_i \alpha_j y_i y_j (\mathbf{x}_i^\top \mathbf{x}_j) , \quad (15.21)$$

$$\text{subject to} \quad \alpha_i \geq 0 , \quad i = 1, \dots, M , \quad (15.22)$$

$$\sum_{i=1}^M \alpha_i y_i = 0 . \quad (15.23)$$

Thus, by solving the dual optimization problem, one obtains the coefficients α_i , $i = 1, \dots, M$, which one needs to express the solution \mathbf{w} . This leads to the decision function:

$$\begin{aligned} f(\mathbf{x}) &= \text{sgn}((\mathbf{w}^\top \mathbf{x}_i) + b) \\ &= \text{sgn}\left(\sum_{i=1}^M y_i \alpha_i (\mathbf{x}_i^\top \mathbf{x}) + b\right) . \end{aligned} \quad (15.24)$$

Note that the scalar product in this dual formulation can be directly replaced by the kernel mapping $k(\mathbf{x}_i, \mathbf{x})$, opening the possibility for the non-linear classifiers. This expression does not directly depend on the dimensionality N of the data but on the number of training examples M . As long as we are able to evaluate the scalar product $(\mathbf{x}_i^\top \mathbf{x})$ the dimensionality could be arbitrary, even infinite.

Non-separable Data

So far we have only considered the separable case which corresponds to an empirical error of zero (cf. Theorem 2). However for many practical applications this assumption is violated. If the data are not linearly separable then problem (15.18) has no feasible solution. By allowing for some errors we might get better results and avoid over-fitting effects (cf. Fig. 15.2).

Therefore a “good” trade-off between the empirical risk and the complexity term in (15.9) needs to be found. Using a technique which was first proposed in (Bennett and Mangasarian, 1992) and later used for SVMs in (Cortes and Vapnik, 1995), one introduces slack-variables to relax the hard-margin constraints:

$$y_i ((\mathbf{w}^\top \mathbf{x}_i) + b) \geq 1 - \xi_i , \quad \xi_i \geq 0 , \quad i = 1, \dots, M , \quad (15.25)$$

additionally allowing for some classification errors. The SVM solution can then be found by (a) keeping the upper bound on the VC-dimension small and (b) by minimizing an upper bound $\sum_{i=1}^M \xi_i$ on the empirical risk, i.e. the number of training errors. Thus, one minimizes

$$\min_{\mathbf{w}, b, \xi} \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^M \xi_i,$$

where the regularization constant $C > 0$ determines the trade-off between the empirical error and the complexity term. This leads to the dual problem:

$$\max_{\alpha} \sum_{i=1}^M \alpha_i - \frac{1}{2} \sum_{i,j=1}^M \alpha_i \alpha_j y_i y_j (\mathbf{x}_i^\top \mathbf{x}_j), \quad (15.26)$$

$$\text{subject to } 0 \leq \alpha_i \leq C, \quad i = 1, \dots, M, \quad (15.27)$$

$$\sum_{i=1}^M \alpha_i y_i = 0. \quad (15.28)$$

From introducing the slack-variables ξ_i , one gets the *box* constraints that limit the size of the Lagrange multipliers: $\alpha_i \leq C, i = 1, \dots, M$.

The threshold b can be computed by exploiting the fact that for all support vectors \mathbf{x}_i with $0 < \alpha_i < C$, the slack variable ξ_i is zero. This follows from the Karush–Kuhn–Tucker (KKT) conditions (cf. (15.29) below). Thus, for any support vector \mathbf{x}_i with $\alpha_i < C$ holds:

$$y_i \left(b + \sum_{j=1}^M y_j \alpha_j (\mathbf{x}_i^\top \mathbf{x}_j) \right) = 1.$$

Averaging over these patterns yields a numerically stable solution:

$$b = \frac{1}{|I|} \sum_{i \in I} \left(y_i - \sum_{j=1}^M y_j \alpha_j (\mathbf{x}_i^\top \mathbf{x}_j) \right),$$

where I denotes the identity matrix.

Sparsity

The Karush–Kuhn–Tucker (KKT) conditions are the necessary conditions for an optimal solution of a non-linear programming problem (e.g. Bertsekas, 1995; Luenberger, 1973). The conditions are particularly simple for the dual SVM problem (??) (Vapnik, 1982):

$$\begin{aligned} \alpha_i = 0 &\Rightarrow y_i f(\mathbf{x}_i) \geq 1 \quad \text{and} \quad \xi_i = 0, \\ 0 < \alpha_i < C &\Rightarrow y_i f(\mathbf{x}_i) = 1 \quad \text{and} \quad \xi_i = 0, \\ \alpha_i = C &\Rightarrow y_i f(\mathbf{x}_i) \leq 1 \quad \text{and} \quad \xi_i \geq 0. \end{aligned} \quad (15.29)$$

They reveal one of the most important properties of SVMs: the solution is sparse in α . For all examples outside the margin area the optimal α_i 's are zero. Specifically,

the KKT conditions show that only such α_i connected to a training pattern \mathbf{x}_i , which is either on the edge of (i.e. $0 < \alpha_i < C$ and $y_i f(\mathbf{x}_i) = 1$) or inside the margin area (i.e. $\alpha_i = C$ and $y_i f(\mathbf{x}_i) < 1$) are non-zero. These are exactly the *support vectors* as mentioned in Sect. 15.3.2. This sparsity property makes SVM learning practical for large data sets.

Decomposition

15.5.2

The practical usefulness of SVM stems from their ability to provide arbitrary non-linear separation boundaries and at the same time to control generalization ability through the parameter C and the kernel parameters. In order to utilize these features it is necessary to work with the dual formulation (15.26)–(15.28) of the SVM training problem. This can be difficult from the computational point of view, for two reasons:

1. One needs to solve the quadratic programming problem with as many variables as the number M of available data points (this can be quite large, up to 10^5 – 10^6).
2. Merely to define the quadratic problem formally, one needs to store the $M \times M$ kernel matrix, which poses an unsurmountable storage problem for large datasets.

Because of these implications, it is usually impossible to use the standard optimization tools (e.g. MINOS, CPLEX, LOQO) for solving the SVM training problems on datasets containing larger than 1000 examples. In the following sections the decomposition techniques are presented, which use the special structure of the SVM problem to provide efficient SVM training algorithms.

Basic Principles

The key idea of decomposition is to freeze all but a small number of optimization variables and to solve a sequence of constant-size problems. The set of variables optimized at a current iteration is referred to as the *working set*. Because the working set is re-optimized, the value of the objective function is improved at each iteration, provided the working set is not optimal before re-optimization.

The mathematics of the decomposition technique can be best seen in the matrix notation. Let $\boldsymbol{\alpha} = (\alpha_1, \dots, \alpha_M)^\top$, let $\mathbf{y} = (y_1, \dots, y_M)^\top$, let H be the matrix with the entries $H_{ij} = y_i y_j k(\mathbf{x}_i, \mathbf{x}_j)$, and let $\mathbf{1}$ denote the vector of length M consisting of all 1s. Then the dual SVM problem (15.26)–(15.28) can be written in the matrix form as:

$$\max_{\boldsymbol{\alpha}} \mathbf{1}^\top \boldsymbol{\alpha} - \frac{1}{2} \boldsymbol{\alpha}^\top H \boldsymbol{\alpha}, \quad (15.30)$$

$$\text{subject to } \mathbf{y}^\top \boldsymbol{\alpha} = 0, \quad (15.31)$$

$$\boldsymbol{\alpha} - C \mathbf{1} \leq 0, \quad (15.32)$$

$$\boldsymbol{\alpha} \geq 0. \quad (15.33)$$

Let us partition the vector α into α_B of the variables to be included in the working set at a current iteration and the vector α_N of the remaining variables. The matrix H is partitioned as

$$H = \begin{bmatrix} H_{BB} & H_{BN} \\ H_{NB} & H_{NN} \end{bmatrix},$$

with the corresponding parts determined by the index sets B and N . By re-writing the problem (15.30)–(15.33) using the partitioned matrix notation, and observing that only the free variables α_B are to be considered as variables, the following sub-problem, to be solved at each iteration, is obtained:

$$\max_{\alpha} (\mathbf{1}_B^\top - \alpha_N^\top H_{NB}) \alpha_B - \frac{1}{2} \alpha_B^\top H_{BB} \alpha_B, \quad (15.34)$$

$$\text{subject to } \mathbf{y}_B^\top \alpha_B = -\mathbf{y}_N^\top \alpha_N, \quad (15.35)$$

$$\alpha_B - C \mathbf{1}_B \leq 0, \quad (15.36)$$

$$\alpha_B \geq 0. \quad (15.37)$$

By choosing the size q of the working set B relatively small (usually $q \leq 100$) one can ensure that the sub-problem (15.34)–(15.37) is easily solved by any optimization tool.

Iteration is carried out until the following termination criteria, derived from Karush–Kuhn–Tucker conditions (15.29), are satisfied to the required precision ε :

$$b - \varepsilon \leq y_i - \sum_{j=1}^M y_j \alpha_j K(\mathbf{x}_i, \mathbf{x}_j) \leq b + \varepsilon, \quad \forall i: 0 < \alpha_i < C, \quad (15.38)$$

$$y_i \left(\sum_{j=1}^M y_j \alpha_j K(\mathbf{x}_i, \mathbf{x}_j) + b \right) \geq 1 - \varepsilon, \quad \forall i: \alpha_i = 0, \quad (15.39)$$

$$y_i \left(\sum_{j=1}^M y_j \alpha_j K(\mathbf{x}_i, \mathbf{x}_j) + b \right) \leq 1 + \varepsilon, \quad \forall i: \alpha_i = C. \quad (15.40)$$

Working Set Selection: Feasible Direction Algorithms

The crucial issue in the decomposition technique presented above is the selection of working sets. First, the provision that a working set must be sub-optimal before re-optimization is essential to prevent the algorithm from cycling. Second, the working set selection affects the speed of the algorithm: if sub-optimal working sets are selected more or less at random, the algorithm converges very slowly. Finally, the working set selection is important in theoretical analysis of decomposition algorithms; in particular in the convergence proofs and in the analysis of the convergence rate.

Two main approaches exist to the working set selection in the SVM decomposition algorithms: the heuristic approach and the feasible direction approach. The former has been used in the original paper of Osuna et al. (1997a) on SVM

decomposition and has been mainly used in the specific flavor of decomposition algorithms called Sequential Minimal Optimization (SMO), presented in the next section. The feasible direction decomposition algorithms root in the original algorithm of Joachims (1999) for pattern recognition (SVM^{light}), with the formal connection to the feasible direction methods of nonlinear programming established by Laskov (2002).

The notion of a “feasible direction” stems from the classical techniques of nonlinear programming subject to linear constraints (Zoutendijk, 1960; Bertsekas, 1995). It refers to the direction along which any step of the magnitude δ satisfying $0 < \delta \leq \delta^0$, for some fixed δ^0 , results in a feasible solution to the nonlinear program. For any nonlinear program, finding the feasible direction amounts to a solution of a linear programming problem. In particular, for the dual SVM training problem (15.26)–(15.28) the following problem must be solved:

$$\max_{\mathbf{d}} \mathbf{g}^\top \mathbf{d}, \quad (15.41)$$

$$\text{subject to } \mathbf{y}^\top \mathbf{d} = 0, \quad (15.42)$$

$$d_i \geq 0, \quad \text{for all } \alpha_i = 0, \quad (15.43)$$

$$d_i \leq 0, \quad \text{for all } \alpha_i = C, \quad (15.44)$$

$$\|\mathbf{d}\|_2 \leq 1, \quad (15.45)$$

where \mathbf{g} is the gradient of the objective function (15.26). To solve the feasible direction problem exactly at each iteration is inefficient because the linear program (15.41)–(15.45) has all M variables. However, an approximate solution to the feasible direction problem can be efficiently found by using the normalization $d_i \in \{-1, 0, 1\}$ and requiring that the number of positive direction components is equal to the number of the negative components. In this case, the solution is obtained by sorting all examples by the quantity $g_i y_i$, and selecting $q/2$ examples with the largest and $q/2$ examples with the smallest values. In fact, by using the heap operations, sorting can be avoided, and the entire selection can be executed in $O(q \log M)$ time. The motivation behind the quantity $g_i y_i$ can be traced back to the first-order Karush–Kuhn–Tucker conditions (Laskov, 2002), which provides the solid formal background for the feasible direction SVM decomposition.

Convergence of the feasible direction SVM decomposition has been proved by Lin (2001), and the linear convergence rate has been observed experimentally (Laskov, 2002).

Sequential Minimal Optimization

The Sequential Minimal Optimization (SMO) algorithm proposed by Platt (1999) is another popular and efficient algorithm for the SVM training. In this algorithm the idea of decomposition is taken to its extreme by reducing the working sets

to two points – the smallest size for which it is possible to maintain the equality constraint (15.28). With two points in the working set, the optimal solution can be computed analytically without calling an optimization tool.

The analytical computation of the optimal solution is based on the following idea: given the solution $(\alpha_1^{\text{old}}, \alpha_2^{\text{old}})$, the optimal update is computed to obtain the solution $(\alpha_1^{\text{new}}, \alpha_2^{\text{new}})$. To carry out the update, first the constraints (15.27)–(15.28) have to be obeyed. The geometry of these constraints depends on whether the labels y_1 and y_2 are equal or not. The two possible configurations are shown in Fig. 15.7. If $y_1 \neq y_2$ (left picture) the solution should be sought along the line $\alpha_1 - \alpha_2 = \gamma$, where $\gamma = \alpha_1^{\text{old}} + y_1 y_2 \alpha_2^{\text{old}}$. If $y_1 = y_2$ (right picture) the solution should be sought along the line $\alpha_1 + \alpha_2 = \gamma$. If the solution transgresses the bound of the box, it should be clipped at the bound.

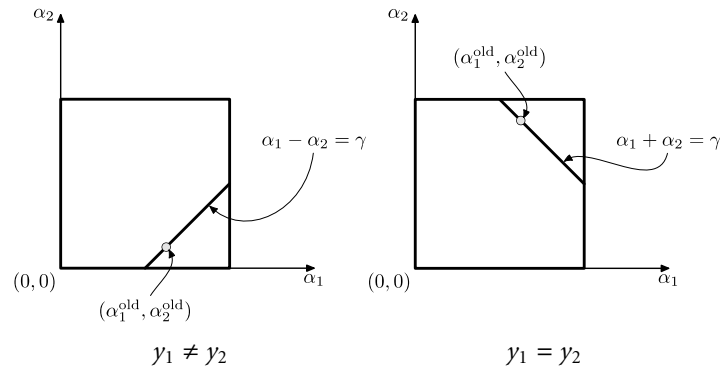


Figure 15.7. Constraints of the SVM training problem with two examples

The optimal values of the variables along the line are computed by eliminating one of the variables through the equality constraint and finding the unconstrained minimum of the objective function. Thus eliminating α_1 one obtains the following update rule for α_2 :

$$\alpha_2^{\text{new}} = \alpha_2^{\text{old}} - \frac{y_2(E_1 - E_2)}{\eta}, \quad (15.46)$$

where

$$E_1 = \sum_{j=1}^M y_j \alpha_j k(\mathbf{x}_1, \mathbf{x}_j) + b - y_1, \quad (15.47)$$

$$E_2 = \sum_{j=1}^M y_j \alpha_j k(\mathbf{x}_2, \mathbf{x}_j) + b - y_2, \quad (15.48)$$

$$\eta = 2k(\mathbf{x}_1, \mathbf{x}_2) - k(\mathbf{x}_1, \mathbf{x}_1) - k(\mathbf{x}_2, \mathbf{x}_2). \quad (15.49)$$

Next, the bound constraints should be taken care of. Depending on the geometry, one computes the following lower and upper bounds on the value of the variable α_2 :

$$L = \begin{cases} \max(0, \alpha_2^{\text{old}} - \alpha_1^{\text{old}}), & \text{if } y_1 \neq y_2, \\ \max(0, \alpha_2^{\text{old}} + \alpha_1^{\text{old}} - C), & \text{if } y_1 = y_2, \end{cases}$$

$$H = \begin{cases} \min(C, C + \alpha_2^{\text{old}} - \alpha_1^{\text{old}}), & \text{if } y_1 \neq y_2, \\ \min(C, \alpha_2^{\text{old}} + \alpha_1^{\text{old}}), & \text{if } y_1 = y_2. \end{cases}$$

The constrained optimum is then found by clipping the unconstrained optimum to the ends of the line segment:

$$\alpha_2^{\text{new}} := \begin{cases} H, & \text{if } \alpha_2^{\text{new}} \geq H, \\ L, & \text{if } \alpha_2^{\text{new}} \leq L, \\ \alpha_2^{\text{new}}, & \text{otherwise.} \end{cases}$$

Finally, the value of α_1^{new} is computed:

$$\alpha_1^{\text{new}} = \alpha_1^{\text{old}} + y_1 y_2 (\alpha_2^{\text{old}} - \alpha_2^{\text{new}}). \quad (15.50)$$

The working set selection in the SMO algorithm is carried out by means of two heuristics. The “first choice” heuristic is responsible for the choice of the first example in each pair. Following this heuristic, all examples that violate the KKT condition (15.29) are used in turns as the first example in the pair. More precisely, the algorithm makes repeated passes only through the examples whose α_i is strictly between then bounds, and only when all such examples satisfy the KKT conditions, the sweep over the entire data is done to bring in new examples. The “second choice” heuristic is responsible for the selection of the second example in the pair. It is intended to bring in such an example that results in the largest step taken during the joint optimization (15.46). As a cheap approximation of this step the numerator $|E_1 - E_2|$ is taken (the denominator, which involves evaluation of kernels, can be expensive to compute). Following the strategy to maximize $|E_1 - E_2|$, the SMO algorithm chooses the example with the largest E_2 if E_1 is negative and the example with the smallest E_2 if E_1 is positive. Under unusual circumstances, when no progress can be made using the second choice heuristic above a hierarchy of weaker heuristics is applied, the details of which can be found in (Platt, 1999).

Incremental Support Vector Optimization

Many real-life machine learning problems can be more naturally viewed as online rather than batch learning problems. Indeed, the data are often collected continuously in time, and, more importantly, the concepts to be learned may also evolve in time. Significant effort has been spent in recent years on the development of the on-line SVM learning algorithms (e.g. Rüping, 2002; Kivinen et al., 2001; Ralaivola and

d'Alché Buc, 2001). An elegant solution to online SVM learning is the incremental SVM proposed by Cauwenberghs and Poggio (2001) which provides a framework for exact online learning.

To present the online SVM learning we first need an abstract formulation of the SVM optimization problem and a brief overview of the incremental SVM. We start with the following abstract form of the SVM optimization problem:

$$\max_{\mu} \min_{\substack{0 \leq \alpha \leq C \\ \mathbf{a}^\top \alpha + b = 0}} : W = -\mathbf{c}^\top \alpha + \frac{1}{2} \alpha^\top K \alpha + \mu (\mathbf{a}^\top \alpha + b) , \quad (15.51)$$

where \mathbf{c} and \mathbf{a} are $M \times 1$ vectors, K is the $M \times M$ kernel matrix and b is a scalar. By defining the meaning of the abstract parameters \mathbf{a} , b and \mathbf{c} for the particular SVM problem at hand, one can use the same algorithmic structure for different SVM algorithms. In particular, for the standard support vector classifiers (Vapnik, 1998), take $\mathbf{c} = \mathbf{1}$, $\mathbf{a} = \mathbf{y}$, $b = 0$ and the given regularization constant C ; the same definition applies to the ν -SVM (Schölkopf et al., 2000) except that $C = 1/(N\nu)$; for the one-class classifier (Schölkopf et al., 2001; Tax and Duin, 2001), the parameters are defined as: $\mathbf{c} = \text{diag}(K)$, $\mathbf{a} = \mathbf{y}$ and $b = -1$.

Incremental SVM provides a procedure for adding one example to an existing optimal solution. When a new point k is added, its weight α_k is initially assigned to 0. Then the weights of other points and μ should be updated, in order to obtain the optimal solution for the enlarged dataset. Likewise, when a point k is to be removed from the dataset, its weight is forced to 0, while updating the weights of the remaining points and μ so that the solution obtained with $\alpha_k = 0$ is optimal for the reduced dataset. The online learning follows naturally from the incremental/decremental learning: the new example is added while some old example is removed from the working set.

The basic principle of the incremental SVM (Cauwenberghs and Poggio, 2001) is that *updates to the state of the example k should keep the remaining examples in their optimal state*. In other words, the KKT conditions (15.29) expressed for the gradients g_i :

$$g_i = -c_i + K_{i,:} \alpha + \mu a_i \begin{cases} \geq 0, & \text{if } \alpha_i = 0, \\ = 0, & \text{if } 0 < \alpha_i < C, \\ \leq 0, & \text{if } \alpha_i = C, \end{cases} \quad (15.52)$$

$$\frac{\partial W}{\partial \mu} = \mathbf{a}^\top \alpha + b = 0, \quad (15.53)$$

must be maintained for all the examples, except possibly for the current example k . Let S denote the set of unbounded support vectors and R the set of other examples. In order to maintain the KKT conditions (15.52), one can express increments to the weights of the unbounded support vectors and to the gradients of other examples as a linear function of the increment of the current example's weight $\Delta \alpha_k$:

$$\Delta \alpha_S = \beta \Delta \alpha_k, \quad \Delta g_R = \gamma \Delta \alpha_k. \quad (15.54)$$

The sensitivity relations (15.54) only make sense for the fixed composition of sets S and R . To proceed with learning, we need to detect the largest increment $\Delta\alpha_k^{\max}$ such that the sets S and R remain intact. After changing the SVM state according to the relations (15.54) evaluated for the increment $\Delta\alpha_k^{\max}$, the sensitivity vectors β and γ must be recomputed accordingly (depending of whether an example enters or leaves the set S). For the details of accounting the reader should refer to (Cauwenberghs and Poggio, 2001; Tax and Laskov, 2003). The iteration terminates when the current element satisfies the KKT conditions (15.52) as well.

Extensions of SVM

15.6

Regression

15.6.1

One extension of the SVM is that for the regression task. In this subsection we will give a short overview of the idea of Support Vector Regression (SVR). A regression problem is given whenever $\mathcal{Y} = \mathbb{R}$ for the training data set $\mathcal{Z} = \{(\mathbf{x}_i, y_i) \in \mathcal{X} \times \mathcal{Y} | i = 1, \dots, M\}$ (cf. Sect. 15.2.1) and our interest is to find a function of the form $f : \mathcal{X} \rightarrow \mathbb{R}$ (or more generally \mathbb{R}^D).

In our discussion of the theoretical foundations of learning we have not yet talked about loss functions except for saying that they should be non-negative functions of the form (15.1). In the following we will discuss an interesting alternative for the problem of regression. There are two loss functions commonly used: the simple squared loss

$$\ell(f(\mathbf{x}), y) = (f(\mathbf{x}) - y)^2, \quad (15.55)$$

and the ε -insensitive loss

$$\ell(f(\mathbf{x}), y) = \begin{cases} |f(\mathbf{x}) - y| - \varepsilon, & \text{if } |f(\mathbf{x}) - y| > \varepsilon, \\ 0, & \text{otherwise.} \end{cases} \quad (15.56)$$

For $\varepsilon = 0$ the ε -insensitive loss equals the ℓ_1 -norm, otherwise it linearly penalizes deviations from the correct predictions by more than ε .

In the left subplot of Fig. 15.8 the two error functions are shown. In the right subplot a regression function using the ε -insensitive loss function is shown for some artificial data. The dashed lines indicate the boundaries of the area where the loss is zero (the “tube”). Clearly most of the data are within the tube.

Similarly to the classification task, one is looking for the function that best describes the values y_i . In classification one is interested in the function that separates two classes; in regression one looks for such a function that contains the given dataset in its ε -tube. Some data points can be allowed to lie outside the ε -tube by introducing the slack-variables.

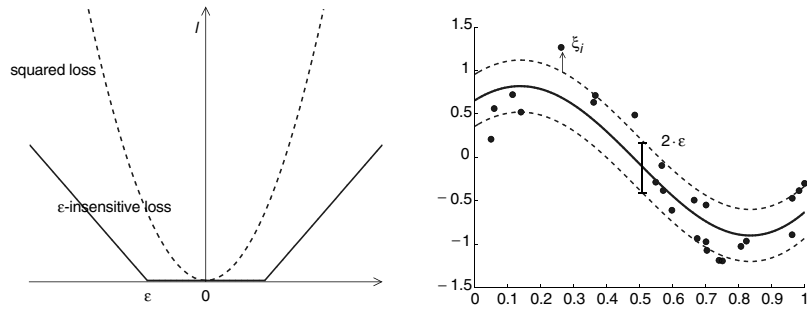


Figure 15.8. The *left subplot* shows the two different loss functions for the regression problem. The *right subplot* gives a regression function derived with an ε -insensitive loss function. The *solid line* indicates the function, the *dashed lines* indicate the ε -tube around this function

The primal formulation for the SVR is then given by:

$$\begin{aligned} \min_{\mathbf{w}, b, \xi, \xi^*} \quad & \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^M (\xi_i + \xi_i^*), \\ \text{subject to} \quad & ((\mathbf{w}^\top \mathbf{x}_i) + b) - y_i \leq \varepsilon + \xi_i, \\ & y_i - ((\mathbf{w}^\top \mathbf{x}_i) + b) \leq \varepsilon + \xi_i^*, \\ & \xi_i, \xi_i^* \geq 0, \quad i = 1, \dots, M. \end{aligned}$$

In contrast to the primal formulation for the classification task, we have to introduce two types of slack-variables ξ_i and ξ_i^* , one to control the error induced by observations that are larger than the upper bound of the ε -tube, and the other – for the observations smaller than the lower bound. To enable the construction of a non-linear regression function, a dual formulation is obtained in the similar way to the classification SVM, and the kernel trick is applied. For an extensive description of SVR we recommend the book of Schölkopf and Smola (2002).

15.6.2 One-Class Classification

Another common problem of statistical learning is one-class classification (novelty detection). Its fundamental difference from the standard classification problem is that the training data is not identically distributed to the test data. The dataset contains two classes: one of them, the target class, is well sampled, while the other class it absent or sampled very sparsely. Schölkopf et al. (2001) have proposed an approach in which the target class is separated from the origin by a hyperplane. Alternatively (Tax and Duin, 2001), the target class can be modeled by fitting a hypersphere with minimal radius around it. We present this approach, schematically shown in Fig. 15.9, in more detail below.

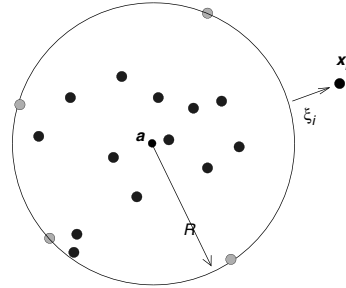


Figure 15.9. Schematic illustration of the hypersphere model for describing a target class of data. The center \mathbf{a} and the radius R should be optimized to minimize the volume of the captured space

Mathematically the problem of fitting a hypersphere around the data is formalized as:

$$\min_{R, \mathbf{a}, \xi} R^2 + C \sum_{i=0}^M \xi_i, \quad (15.57)$$

subject to $\|\mathbf{x}_i - \mathbf{a}\|^2 \leq R^2 + \xi_i, \quad i = 1, \dots, M,$

$$\xi_i \geq 0,$$

where \mathbf{a} is the center of the sphere, and R is the radius. Similarly to the SVM we make a “soft” fit by allowing non-negative slacks ξ_i . One can likewise apply the kernel trick by deriving the dual formulation of (15.57):

$$\max_{\alpha} \sum_{i=1}^M \alpha_i k(\mathbf{x}_i, \mathbf{x}_i) - \sum_{i,j=1}^M \alpha_i \alpha_j k(\mathbf{x}_i, \mathbf{x}_j), \quad (15.58)$$

subject to $0 \leq \alpha_i \leq C, \quad i = 1, \dots, M,$

$$\sum_{i=1}^M \alpha_i = 1.$$

The parameter C can be interpreted (Schölkopf et al., 2001) as the reciprocal of the quantity $\frac{1}{M\nu}$, where ν is an upper bound for the fraction of objects outside the boundary.

To decide whether a new object belongs to the target class one should determine its position with respect to the sphere using the formula

$$f(\mathbf{x}) = \text{sign} \left(R^2 - k(\mathbf{x}, \mathbf{x}) + 2 \sum_i \alpha_i k(\mathbf{x}, \mathbf{x}_i) - \sum_{i,j} \alpha_i \alpha_j k(\mathbf{x}_i, \mathbf{x}_j) \right). \quad (15.59)$$

An object with positive sign belongs to the target class and vice versa.

15.7 Applications

15.7.1 Optical Character Recognition (OCR)

One of the first real-world experiments carried out with SVM was done at AT&T Bell Labs using optical character recognition (OCR) data (Cortes and Vapnik, 1995; Schölkopf et al., 1995). These early experiments already showed the astonishingly high accuracies for SVMs which was on a par with convolutive multi-layer perceptrons. Below we record the classification performance of SVM, some variants not discussed in this chapter, and other classifiers on the USPS (US-Postal Service) benchmark (parts from Simard et al., 1998). The task is to classify handwritten digits into one of ten classes. Standard SVMs as presented here already exhibit a performance similar to other methods.

Table 15.2. Classification results on the USPS data set

Linear PCA & Linear SVM (Schölkopf et al., 1998b)	8.7%
k-Nearest Neighbor	5.7%
LeNet1 (Bottou et al., 1994)	4.2%
Regularised RBF Networks (Rätsch, 1998)	4.1%
Kernel-PCA & linear SVM (Schölkopf et al., 1998b)	4.0%
SVM (Schölkopf et al., 1995)	4.0%
Invariant SVM (Schölkopf et al., 1998a)	3.0%
Boosting (Drucker et al., 1993)	2.6%
Tangent Distance (Simard et al., 1998)	2.5%
Human error rate	2.5%

A benchmark problem larger than the USPS data set (7291 patterns) was collected by NIST and contains 120,000 handwritten digits. Invariant SVMs achieve an error rate of 0.7% (DeCoste and Schölkopf, 2002) on this challenging and more realistic data set, slightly better than the tangent distance method (1.1%) or single neural networks (LeNet 5: 0.9%). An ensemble of LeNet 4 networks that was trained on a vast number of artificially generated patterns (using invariance transformations) is on a par with the best SVM and also achieved 0.7% (LeCun et al., 1995).

15.7.2 Text Categorization and Text Mining

The high dimensional problem of text categorization seems to be an application for which SVMs have performed particularly well. A popular benchmark is the Reuters-22,173 text corpus, where Reuters collected 21,450 news stories from 1997, and partitioned and indexed them into 135 different categories to simplify the access. The feature typically used to classify Reuters documents are 10,000-dimensional vectors containing word frequencies within a document. With such a coding SVMs have achieved excellent results (see e.g. Joachims, 1997).

Active Learning in Drug Design

15.7.3

A challenging problem of computational chemistry is the discovery of active chemical compounds. The goal is to find the compounds that bind to a certain molecule in as few iterations of biological testing as possible. An astonishingly effective application of SVM to this problem has been recently proposed by Warmuth et al. (2003).

Other Applications

15.7.4

There are numerous other applications to which SVM were successfully applied. Examples are object and face recognition tasks (Osuna et al., 1997b), inverse problems (Vapnik, 1998; Weston et al., 1999), gene array expression monitoring (Brown et al., 2000), remote protein homology detection (Jaakkola et al., 2000), or splice site detection (Zien et al., 2000; Sonnenburg et al., 2002). This list can be extended.

Summary and Outlook

15.8

We have seen how the problem of learning from data can be cast formally into the problem of estimating functions from given observations. We reviewed some basic notation and concepts from statistics and especially from statistical learning theory. The latter provides us with two extremely important insights: (1) what matter the most is not the dimensionality of the data but the complexity of the function class we choose our estimate from, (2) consistency is desirable for successful learning. Closely related to these two insights is the issue of regularization. Regularization allows us to *control* the complexity of our learning machine and often suffices to achieve consistency.

As an application of statistical learning theory we reviewed maximum margin hyperplanes. Whilst it is satisfactory to have a technique at hand that implements (at least partially) what the theory justifies, the algorithm is only capable of finding (linear) hyperplanes. To circumvent this restriction we introduced kernel functions yielding SVMs. Kernel functions allow us to reformulate many algorithms in some kernel feature space that is non-linearly related to the input space and yield powerful, non-linear techniques. This non-linearization using the kernel trick is possible whenever we are able to express an algorithm such that it only uses the data in the form of scalar products. However, since the algorithms are still linear in the feature space we can use the same theory and optimization strategies as before.

Kernel algorithms have seen an extensive development over the past years, starting with the SVM. Among many theoretical (Williamson et al., 1998; Graepel et al., 2000; Bartlett et al., 2002) and algorithmic (Platt, 1999; Joachims, 1999) results on SVM itself, new algorithms using the kernel trick have been proposed (e.g. Kernel PCA (Schölkopf et al., 1998b) or Bayes-Point machines (Herbrich et al., 2001)). This development is still an ongoing and exciting field of study.

To conclude, we would like to encourage the reader to follow the presented methodology of (re-)formulating linear algorithms based on scalar product into nonlinear algorithms using the powerful kernel trick, to further develop statistical learning techniques. Information on recent developments in kernel methods can be found at <http://www.kernel-machines.org/>.

References

- Aizerman, M., Braverman, E. and Rozonoer, L. (1964). Theoretical foundations of the potential function method in pattern recognition learning. *Automation and Remote Control*, 25: 821–837.
- Akaike, H. (1974). A new look at the statistical model identification. *IEEE Trans. Automat. Control*, 19(6): 716–723.
- Aronszajn, N. (1950). Theory of reproducing kernels. *Transactions of the American Mathematical Society*, 68: 337–404.
- Barron, A., Birgé, L. and Massart, P. (1999). Risk bounds for model selection via penalization. *Probability Theory and Related Fields*, 113: 301–415.
- Bartlett, P., Bousquet, O. and Mendelson, S. (2002). Localized rademacher complexities. In Kivinen, J. and Sloan, R.H., (eds), *Proceedings COLT*, volume 2375 of *Lecture Notes in Computer Science*, pages 44–58. Springer, Heidelberg.
- Bartlett, P.L., Long, P.M. and Williamson, R.C. (1996). Fat-shattering and the learnability of real-valued functions. *Journal of Computer and System Sciences*, 52(3): 434–452.
- Bartlett, P.L. and Mendelson, S. (2002). Rademacher and gaussian complexities: Risk bounds and structural results. *Journal of Machine Learning Research*, 3: 463–482.
- Bennett, K.P. and Mangasarian, O.L. (1992). Robust linear programming discrimination of two linearly inseparable sets. *Optimization Methods and Software*, 1: 23–34.
- Bertsekas, D.P. (1995). *Nonlinear Programming*. Athena Scientific, Belmont, MA.
- Bishop, C.M. (1995). *Neural Networks for Pattern Recognition*. Oxford University Press.
- Blankertz, B., Curio, G. and Müller, K-R. (2002). Classifying single trial EEG: Towards brain computer interfacing. In Diettrich, T.G., Becker, S., and Ghahramani, Z., (eds), *Advances in Neural Information Processing Systems*, 14: 157–164.
- Boser, B.E., Guyon, I.M. and Vapnik, V.N. (1992). A training algorithm for optimal margin classifiers. In Haussler, D. (ed), *Proceedings of the 5th Annual ACM Workshop on Computational Learning Theory*, pp.144–152.
- Bottou, L., Cortes, C., Denker, J.S., Drucker, H., Guyon, I., Jackel, L.D., LeCun, Y.A., Müller, U.A., Säckinger, E., Simard, P.Y. and Vapnik, V.N. (1994). Comparison of classifier methods: a case study in handwritten digit recognition. In *Proceedings of the 12th International Conference on Pattern Recognition and Neural Networks, Jerusalem*, pp.77–87, IEEE Computer Society Press.

- Breiman, L., Friedman, J., Olshen, J. and Stone, C. (1984). *Classification and Regression Trees*, Wadsworth.
- Brown, M.P.S., Grundy, W.N., Lin, D., Cristianini, N., Sugnet, C., Furey, T.S., Ares, M. and Haussler, D. (2000). Knowledge-based analysis of microarray gene expression data using support vector machines. *Proceedings of the National Academy of Sciences*, 97(1): 262–267.
- Cauwenberghs, G. and Poggio, T. (2001). Incremental and decremental support vector machine learning. In Leen, T.K., Dietterich, T.G. and Tresp, V. (eds), *Advances in Neural Information Processing Systems*, 13: 409–415, MIT Press.
- Cortes, C. and Vapnik, V.N. (1995). Support vector networks. *Machine Learning*, 20: 273–297.
- Cristianini, N. and Shawe-Taylor, J. (2000) *An Introduction to Support Vector Machines*, Cambridge University Press, Cambridge, UK.
- DeCoste, D. and Schölkopf, B. (2002) Training invariant support vector machines. *Machine Learning*, 46: 161–190. Also: Technical report JPL-MLTR-00-1, Jet Propulsion Laboratory, Pasadena.
- Devroye, L., Györfi, L. and Lugosi, G. (1996). *A Probabilistic Theory of Pattern Recognition*. Number 31 in Applications of Mathematics. Springer, New York.
- Donoho, D., Johnstone, I., Kerkycharian, G. and Picard, D. (1996). Density estimation by wavelet thresholding. *Annals of Statistics*, 24: 508–539.
- Drucker, H., Schapire, R. and Simard, P.Y. (1993). Boosting performance in neural networks. *International Journal of Pattern Recognition and Artificial Intelligence*, 7: 705–719.
- Duda, R.O., Hart, P.E. and Stork, D.G. (2001). *Pattern classification*, John Wiley & Sons, second edition.
- Freund, Y. and Schapire, R.E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1): 119–139.
- Girosi, F. (1998). An equivalence between sparse approximation and support vector machines. *Neural Computation*, 10: 1455–1480.
- Girosi, F., Jones, M. and Poggio, T. (1993). Priors, stabilizers and basis functions: From regularization to radial, tensor and additive splines. Technical Report A.I. Memo No. 1430, Massachusetts Institute of Technology.
- Graepel, T., Herbrich, R. and Shawe-Taylor, J. (2000). Generalization error bounds for sparse linear classifiers. In *Proc. COLT*, pp. 298–303, Morgan Kaufmann, San Francisco.
- Haussler, D. (1999). Convolution kernels on discrete structures. Technical Report UCSC-CRL-99-10, UC Santa Cruz.
- Herbrich, R., Graepel, T. and Campbell, C. (2001). Bayes point machines. *Journal of Machine Learning Research*, 1: 245–279.
- Jaakkola, T.S., Diekhans, M. and Haussler, D. (2000). A discriminative framework for detecting remote protein homologies. *J. Comp. Biol.*, 7: 95–114.
- Joachims, T. (1997). Text categorization with support vector machines: Learning with many relevant features. Technical Report 23, LS VIII, University of Dortmund.

- Joachims, T. (1999). Making large-scale SVM learning practical. In Schölkopf, B., Burges, C.J.C. and Smola, A.J. (eds), *Advances in Kernel Methods – Support Vector Learning*, pp. 169–184, Cambridge, MA, MIT Press.
- Kivinen, J., Smola, A.J. and Williamson, R.C. (2001). Online learning with kernels. In Diettrich, T.G., Becker, S. and Ghahramani, Z. (eds), *Advances in Neural Inf. Proc. Systems (NIPS 01)*, pp. 785–792.
- Kolmogorov, A.N. (1941). Stationary sequences in hilbert spaces. *Moscow University Mathematics*, 2: 1–40.
- Laskov, P. (2002). Feasible direction decomposition algorithms for training support vector machines. *Machine Learning*, 46: 315–349.
- LeCun, Y.A., Jackel, L.D., Bottou, L., Brunot, A., Cortes, C., Denker, J.S., Drucker, H., Guyon, I., Müller, U.A., Säckinger, E., Simard, P.Y. and Vapnik, V.N. (1995). Comparison of learning algorithms for handwritten digit recognition. In Fogelman-Soulié, F. and Gallinari, P. (eds), *Proceedings ICANN'95 – International Conference on Artificial Neural Networks*, II: 53–60, Nanterre, France, EC2.
- Lin, C.-J. (2001). On the convergence of the decomposition method for support vector machines. *IEEE Trans. on Neural Networks*, 12(6): 1288–1298.
- Luenberger, D.G. (1973). *Introduction to Linear and Nonlinear Programming*. Addison-Wesley, Reading, MA.
- Mallows, C.L. (1973). Some comments on cp. *Technometrics*, 15: 661–675.
- Mercer, J. (1909). Functions of positive and negative type and their connection with the theory of integral equations. *Philos. Trans. Roy. Soc. London*, A 209: 415–446.
- Mika, S. (2002). *Kernel Fisher Discriminants*. PhD thesis, University of Technology, Berlin, Germany.
- Moody, J. and Darken, C. (1989). Fast learning in networks of locally-tuned processing units. *Neural Computation*, 1(2): 281–294.
- Morozov, V.A. (1984). *Methods for Solving Incorrectly Posed Problems*. Springer.
- Müller, K.-R., Mika, S., Rätsch, G., Tsuda, K. and Schölkopf, B. (2001). An introduction to kernel-based learning algorithms. *IEEE Transactions on Neural Networks*, 12(2): 181–201.
- Osuna, E., Freund, R. and Girosi, F. (1997a). An improved training algorithm for support vector machines. In Principe, J., Gile, L., Morgan, N. and Wilson, E. (eds), *Neural Networks for Signal Processing VII – Proceedings of the 1997 IEEE Workshop*, pp. 276–285, New York, IEEE.
- Osuna, E., Freund, R. and Girosi, F. (1997b). Training support vector machines: An application to face detection. In *Proceedings CVPR'97*.
- Parzen, E. (1962). On estimation of probability density function and mode. *Annals of Mathematical Statistics*, 33: 1065–1076.
- Platt, J. (1999). Fast training of support vector machines using sequential minimal optimization. In Schölkopf, B., Burges, C.J.C. and Smola, A.J. (eds), *Advances in Kernel Methods – Support Vector Learning*, pp. 185–208, MIT Press, Cambridge, MA.

- Ralaivola, L. and d'Alché Buc, F. (2001). Incremental support vector machine learning: A local approach. *Lecture Notes in Computer Science*, 2130: 322–329, URL citeseer.nj.nec.com/ralaivolaoiincremental.html.
- Rätsch, G. (1998). Ensemble learning methods for classification. Master's thesis, Dep. of Computer Science, University of Potsdam, Germany.
- Rätsch, G. (2001). *Robust Boosting via Convex Optimization*. PhD thesis, University of Potsdam, Neues Palais 10, 14469 Potsdam, Germany.
- Rätsch, G., Mika, S., Schölkopf, B. and Müller, K.-R. (2002). Constructing boosting algorithms from SVMs: an application to one-class classification. *IEEE PAMI*, 24(9): 1184–1199. Earlier version is GMD TechReport No. 119 (2000).
- Rüping, S. (2002). Incremental learning with support vector machines. Technical Report TR-18, Universität Dortmund, SFB475.
- Schölkopf, B., Burges, C.J.C. and Vapnik, V.N. (1995). Extracting support data for a given task. In Fayyad, U.M. and Uthurusamy, R. (eds), *Proceedings, First International Conference on Knowledge Discovery & Data Mining*, AAAI Press, Menlo Park, CA.
- Schölkopf, B. (2001). The kernel trick for distances. In Leen, T.K., Diettrich, T.G. and Tresp, V. (eds), *Advances in Neural Information Processing Systems 13*. MIT Press.
- Schölkopf, B., Platt, J., Shawe-Taylor, J., Smola, A.J. and Williamson, R.C. (2001). Estimating the support of a high-dimensional distribution. *Neural Computation*, 13(7): 1443–1471.
- Schölkopf, B., Simard, P.Y., Smola, A.J. and Vapnik, V.N. (1998a). Prior knowledge in support vector kernels. In Jordan, M., Kearns, M. and Solla, S. (eds), *Advances in Neural Information Processing Systems*, 10: 640–646, MIT Press, Cambridge, MA.
- Schölkopf, B., Smola, A., Williamson, R.C. and Bartlett, P.L. (2000). New support vector algorithms. *Neural Computation*, 12: 1207–1245. also NeuroCOLT Technical Report NC-TR-1998-031.
- Schölkopf, B. and Smola, A.J. (2002). *Learning with Kernels*. MIT Press, Cambridge, MA.
- Schölkopf, B., Smola, A.J. and Müller, K.-R. (1998b). Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10: 1299–1319.
- Shawe-Taylor, J., Bartlett, P.L. and Williamson, R.C. (1998) Structural risk minimization over data-dependent hierarchies. *IEEE Transactions on Information Theory*, 44(5): 1926–1940.
- Simard, P.Y., LeCun, Y.A., Denker, J.S. and Victorri, B. (1998). Transformation invariance in pattern recognition – tangent distance and tangent propagation. In Orr, G. and Müller, K.-R. (eds), *Neural Networks: Tricks of the Trade*, LNCS 1524: 239–274. Springer.
- Smola, A.J., Schölkopf, B. and Müller, K.-R. (1998). The connection between regularization operators and support vector kernels. *Neural Networks*, 11: 637–649.
- Sonnenburg, S., Rätsch, G., Jagota, A. and Müller, K.-R. (2002). New methods for splice-site recognition. In Dorronsoro, J.R. (ed), *Proc. International conference*

- on artificial Neural Networks – ICANN’02, pp. 329–336, LNCS 2415, Springer, Berlin.
- Stitson, M., Gammerman, A., Vapnik, V.N., Vovk, V., Watkins, C. and Weston, J. (1997). Support vector regression with ANOVA decomposition kernels. Technical Report CSD-97-22, Royal Holloway, University of London.
- Tax, D. and Laskov, P. (2003). Online SVM learning: from classification to data description and back. In Molina, C. et al. (ed), *Proc. NNSP*, pp.499–508.
- Tax, D.M.J. and Duin, R.P.W. (2001). Uniform object generation for optimizing one-class classifiers. *Journal for Machine Learning Research*, pp. 155–173.
- Tikhonov, A.N. and Arsenin, V.Y. (1977). *Solutions of Ill-posed Problems*. W.H. Winston, Washington, D.C.
- Tsuda, K., Kawanabe, M., Rätsch, G., Sonnenburg, S. and Müller, K.R. (2002). A new discriminative kernel from probabilistic models. *Neural Computation*, 14: 2397–2414.
- Vapnik, V.N. (1982). *Estimation of Dependences Based on Empirical Data*. Springer, Berlin.
- Vapnik, V.N. (1998). *Statistical Learning Theory*. Wiley, New York.
- Vapnik, V.N. and Chervonenkis, A.Y. (1974). *Theory of Pattern Recognition*. Nauka, Moscow, Russian.
- Vapnik, V.N. and Chervonenkis, A.Y. (1991). The necessary and sufficient conditions for consistency in the empirical risk minimization method. *Pattern Recognition and Image Analysis*, 1(3): 283–305.
- Wahba, G. (1980). Spline bases, regularization, and generalized cross-validation for solving approximation problems with large quantities of noisy data. In *Proceedings of the International Conference on Approximation theory*. Academic Press, Austin, Texas.
- Warmuth, M.K, Liao, J., Rätsch, G., Mathieson, M., Putta, S. and Lemmem, C. (2003). Support Vector Machines for active learning in the drug discovery process. *Journal of Chemical Information Sciences*, 43(2): 667–673.
- Watkins, C. (2000). Dynamic alignment kernels. In Smola, A.J., Bartlett, P.L., Schölkopf, B. and Schuurmans, D. (eds), *Advances in Large Margin Classifiers*, pp.39–50, MIT Press, Cambridge, MA.
- Weston, J., Gammerman, A., Stitson, M., Vapnik, V.N., Vovk, V. and Watkins, C. (1999). Support vector density estimation. In Schölkopf, B., Burges, C.J.C. and Smola, A.J. (eds), *Advances in Kernel Methods – Support Vector Learning*, pp. 293–305, MIT Press, Cambridge, MA.
- Williamson, R.C., Smola, A.J. and Schölkopf, B. (1998). Generalization performance of regularization networks and support vector machines via entropy numbers of compact operators. NeuroCOLT Technical Report NC-TR-98-019, Royal Holloway College, University of London, UK.
- Zien, A., Rätsch, G., Mika, S., Schölkopf, B., Lengauer, T. and Müller, K.-R. (2000). Engineering Support Vector Machine Kernels That Recognize Translation Initiation Sites. *Bioinformatics*, 16(9): 799–807.
- Zoutendijk, G. (1960). *Methods of feasible directions*, Elsevier.

Bagging, Boosting and Ensemble Methods

III.16

Peter Bühlmann

16.1	<i>An Introduction to Ensemble Methods</i>	878
16.2	<i>Bagging and Related Methods</i>	878
	Bagging	879
	Unstable Estimators with Hard Decision Indicator	881
	Subbagging	884
	Bagging More “Smooth” Base Procedures and Bragging	885
	Bragging	886
	Out-of-Bag Error Estimation	886
	Disadvantages	886
	Other References	887
16.3	<i>Boosting</i>	887
	Boosting as Functional Gradient Descent	888
	The Generic Boosting Algorithm	890
	Small Step Size	894
	The Bias-variance Trade-off for L_2 Boost	894
	L_2 Boost with Smoothing Spline Base Procedure for One-dimensional Curve Estimation	895
	L_2 Boost for Additive and Interaction Regression Models	896
	Linear Modeling	903
	Boosting Trees	903
	Boosting and ℓ_1 -penalized Methods (Lasso)	904
	Other References	904

An Introduction to Ensemble Methods

Ensemble methods aim at improving the predictive performance of a given statistical learning or model fitting technique. The general principle of ensemble methods is to construct a linear combination of some model fitting method, instead of using a single fit of the method.

More precisely, consider for simplicity the framework of function estimation. We are interested in estimating a real-valued function

$$g : \mathbb{R}^d \rightarrow \mathbb{R}$$

based on data $(X_1, Y_1), \dots, (X_n, Y_n)$ where X is a d -dimensional predictor variable and Y a univariate response. Generalizations to other functions $g(\cdot)$ and other data-types are possible. We assume to have specified a *base procedure* which, given some input data (as above), yields an estimated function $\hat{g}(\cdot)$. For example, the base procedure could be a nonparametric kernel estimator (if d is small) or a nonparametric statistical method with some structural restrictions (for $d \geq 2$) such as a regression tree (or class-probability estimates from a classification tree). We can run a base procedure many times when changing the input data: the original idea of ensemble methods is to use reweighted original data to obtain different estimates $\hat{g}_1(\cdot), \hat{g}_2(\cdot), \hat{g}_3(\cdot), \dots$ based on different reweighted input data. We can then construct an ensemble-based function estimate $g_{ens}(\cdot)$ by taking linear combinations of the individual function estimates $\hat{g}_k(\cdot)$:

$$\hat{g}_{ens}(\cdot) = \sum_{k=1}^M c_k \hat{g}_k(\cdot), \quad (16.1)$$

where the $\hat{g}_k(\cdot)$ are obtained from the base procedure based on the k th reweighted data-set. For some ensemble methods, e.g. for bagging (see Sect. 16.2), the linear combination coefficients $c_k \equiv 1/M$ are averaging weights; for other methods, e.g. for boosting (see Sect. 16.3), $\sum_{k=1}^M c_k$ increases as M gets larger.

Ensemble methods became popular as a relatively simple device to improve the predictive performance of a base procedure. There are different reasons for this: the bagging procedure turns out to be a variance reduction scheme, at least for some base procedures. On the other hand, boosting methods are primarily reducing the (model) bias of the base procedure. This already indicates that bagging and boosting are very different ensemble methods. We will argue in Sects. 16.3.1 and 16.3.6 that boosting may be even viewed as a non-ensemble method which has tremendous advantages over ensemble (or multiple prediction) methods in terms of interpretation.

Random forests (Bre99b) is a very different ensemble method than bagging or boosting. The earliest random forest proposal is from Amit and Geman (AG97). From the perspective of prediction, random forests is about as good as boosting, and often better than bagging. For further details about random forests we refer to (Bre99b).

Some rather different exposition about bagging and boosting which describes these methods in the much broader context of many other modern statistical methods can be found in (HTF01).

Bagging and Related Methods

16.2

Bagging (Bre96a), a sobriquet for bootstrap aggregating, is an ensemble method for improving unstable estimation or classification schemes. Breiman (Bre96a) motivated bagging as a variance reduction technique for a given base procedure, such as decision trees or methods that do variable selection and fitting in a linear model. It has attracted much attention, probably due to its implementational simplicity and the popularity of the bootstrap methodology. At the time of its invention, only heuristic arguments were presented why bagging would work. Later, it has been shown in (BY02) that bagging is a smoothing operation which turns out to be advantageous when aiming to improve the predictive performance of regression or classification trees. In case of decision trees, the theory in (BY02) confirms Breiman's intuition that bagging is a variance reduction technique, reducing also the mean squared error (MSE). The same also holds for subbagging (subsample aggregating), defined in Sect. 16.2.3, which is a computationally cheaper version than bagging. However, for other (even "complex") base procedures, the variance and MSE reduction effect of bagging is not necessarily true; this has also been shown in (BS02) for the simple case where the estimator is a U -statistics.

Bagging

16.2.1

Consider the regression or classification setting. The data is given as in Sect. 16.1: we have pairs (X_i, Y_i) ($i = 1, \dots, n$), where $X_i \in \mathbb{R}^d$ denotes the d -dimensional predictor variable and the response $Y_i \in \mathbb{R}$ (regression) or $Y_i \in \{0, 1, \dots, J-1\}$ (classification with J classes). The target function of interest is usually $\mathbb{E}[Y|X = x]$ for regression or the multivariate function $\mathbb{P}[Y = j|X = x]$ ($j = 0, \dots, J-1$) for classification. The function estimator, which is the result from a given base procedure, is

$$\hat{g}(\cdot) = h_n((X_1, Y_1), \dots, (X_n, Y_n))(\cdot) : \mathbb{R}^d \rightarrow \mathbb{R},$$

where the function $h_n(\cdot)$ defines the estimator as a function of the data.

Bagging is defined as follows.

Bagging algorithm

Step 1. Construct a bootstrap sample $(X_1^*, Y_1^*), \dots, (X_n^*, Y_n^*)$ by randomly drawing n times with replacement from the data $(X_1, Y_1), \dots, (X_n, Y_n)$.

Step 2. Compute the bootstrapped estimator $\hat{g}^*(\cdot)$ by the plug-in principle: $\hat{g}^*(\cdot) = h_n((X_1^*, Y_1^*), \dots, (X_n^*, Y_n^*))(\cdot)$.

Step 3. Repeat steps 1 and 2 M times, where M is often chosen as 50 or 100, yielding $\hat{g}^{*k}(\cdot)$ ($k = 1, \dots, M$). The bagged estimator is $\hat{g}_{\text{Bag}}(\cdot) = M^{-1} \sum_{k=1}^M \hat{g}^{*k}(\cdot)$.

In theory, the bagged estimator is

$$\hat{g}_{\text{Bag}}(\cdot) = \mathbb{E}^* [\hat{g}^*(\cdot)] . \quad (16.2)$$

The theoretical quantity in (16.2) corresponds to $M = \infty$: the finite number M in practice governs the accuracy of the Monte Carlo approximation but otherwise, it shouldn't be viewed as a tuning parameter for bagging. Whenever we discuss properties of bagging, we think about the theoretical version in (16.2).

This is exactly Breiman's (Bre96a) definition for bagging regression estimators. For classification, we propose to average the bootstrapped probabilities $\hat{g}_j^{*k}(\cdot) = \mathbb{P}^*[Y^{*k} = j | X^{*k} = \cdot]$ ($j = 0, \dots, J - 1$) yielding an estimator for $\mathbb{P}[Y = j | X = \cdot]$, whereas Breiman (Bre96a) proposed to vote among classifiers for constructing the bagged classifier.

The empirical fact that bagging improves the predictive performance of regression and classification trees is nowadays widely documented (Bre96a, Bre96b, BY02, BDO2, BSO2, HLO2). To give an idea about the gain in performance, we cite some of the results of Breiman's pioneering paper (Bre96a): for 7 classification problems, bagging a classification tree improved over a single classification tree (in terms of cross-validated misclassification error) by

33%, 47%, 30%, 23%, 20%, 22%, 27% ;

in case of 5 regression data sets, bagging regression trees improved over a single regression tree (in terms of cross-validated squared error) by

39%, 22%, 46%, 30%, 38% .

In both cases, the size of the single decision tree and of the bootstrapped trees was chosen by optimizing a 10-fold cross-validated error, i.e. using the "usual" kind of tree procedure. Besides that the reported improvement in percentages is quite impressive, it is worth pointing out that bagging a decision tree is almost never worse (in terms of predictive power) than a single tree.

A trivial equality indicates the somewhat unusual approach of using the bootstrap methodology:

$$\hat{g}_{\text{Bag}}(\cdot) = \hat{g}(\cdot) + (\mathbb{E}^*[\hat{g}^*(\cdot)] - \hat{g}(\cdot)) = \hat{g}(\cdot) + \text{Bias}^*(\cdot) ,$$

where $\text{Bias}^*(\cdot)$ is the bootstrap bias estimate of $\hat{g}(\cdot)$. Instead of the usual bias correction with a negative sign, bagging comes along with the wrong sign and *adds* the bootstrap bias estimate. Thus, we would expect that bagging has a higher bias than $\hat{g}(\cdot)$, which we will argue to be true in some sense, see Sect. 16.2.2. But according to the usual interplay between bias and variance in nonparametric statistics, the hope is to gain more by reducing the variance than increasing the bias, so that overall, bagging would pay-off in terms of the MSE. Again, this hope

turns out to be true for some base procedures. In fact, Breiman (Bre96a) described heuristically the performance of bagging as follows: the variance of the bagged estimator $\hat{g}_{\text{Bag}}(\cdot)$ should be equal or smaller than that for the original estimator $\hat{g}(\cdot)$; and there can be a drastic variance reduction if the original estimator is “unstable”.

Unstable Estimators with Hard Decision Indicator

16.2.2

Instability often occurs when hard decisions with indicator functions are involved as in regression or classification trees. One of the main underlying ideas why bagging works can be demonstrated by a simple example.

Example 1: Toy Example: A Simple, Instructive Analysis

1

Consider the estimator

$$\hat{g}(x) = \mathbf{1}_{[\bar{Y}_n \leq x]}, \quad x \in \mathbb{R}, \quad (16.3)$$

where $\bar{Y}_n = n^{-1} \sum_{i=1}^n Y_i$ with Y_1, \dots, Y_n i.i.d. (no predictor variables X_i are used for this example). The target we have in mind is $g(x) = \lim_{n \rightarrow \infty} \mathbb{E}[\hat{g}(x)]$. A simple yet precise analysis below shows that bagging is a smoothing operation. Due to the central limit theorem we have

$$n^{1/2}(\bar{Y}_n - \mu) \rightarrow_D \mathcal{N}(0, \sigma^2) \quad (n \rightarrow \infty) \quad (16.4)$$

with $\mu = \mathbb{E}[Y_1]$ and $\sigma^2 = \text{Var}(Y_1)$. Then, for x in a $n^{-1/2}$ -neighborhood of μ ,

$$x = x_n(c) = \mu + c\sigma n^{-1/2}, \quad (16.5)$$

we have the distributional approximation

$$\hat{g}(x_n(c)) \rightarrow_D L(Z) = \mathbf{1}_{[Z \leq c]} \quad (n \rightarrow \infty), \quad Z \sim \mathcal{N}(0, 1). \quad (16.6)$$

Obviously, for a fixed c , this is a hard decision function of Z . On the other hand, averaging for the bagged estimator looks as follows. Denote by $\Phi(\cdot)$ the c.d.f. of a standard normal distribution:

$$\begin{aligned} \hat{g}_{\text{Bag}}(x_n(c)) &= \mathbb{E}^* \left[\mathbf{1}_{[\bar{Y}_n^* \leq x_n(c)]} \right] = \mathbb{E}^* \left[\mathbf{1}_{[n^{1/2}(\bar{Y}_n^* - \bar{Y}_n)/\sigma \leq n^{1/2}(x_n(c) - \bar{Y}_n)/\sigma]} \right] \\ &= \Phi \left(n^{1/2}(x_n(c) - \bar{Y}_n)/\sigma \right) + o_p(1) \\ &\rightarrow_D L_{\text{Bag}}(Z) = \Phi(c - Z) \quad (n \rightarrow \infty), \quad Z \sim \mathcal{N}(0, 1), \end{aligned} \quad (16.7)$$

where the first approximation (second line) follows because the bootstrap works for the arithmetic mean \bar{Y}_n , i.e.,

$$\sup_{x \in \mathbb{R}} \left| \mathbb{P}^* \left[n^{1/2}(\bar{Y}_n^* - \bar{Y}_n)/\sigma \leq x \right] - \Phi(x) \right| = o_p(1) \quad (n \rightarrow \infty), \quad (16.8)$$

and the second approximation (third line in (16.7) holds, because of (16.4) and the definition of $x_n(c)$ in (16.5). Comparing with (16.6), bagging produces a soft decision function $L_{\text{Bag}}(\cdot)$ of Z : it is a shifted inverse probit, similar to a sigmoid-type function. Figure 16.1 illustrates the two functions $L(\cdot)$ and $L_{\text{Bag}}(\cdot)$. We see that bagging is a smoothing operation. The amount of smoothing is determined “automatically” and turns out to be very reasonable (we are not claiming any optimality here). The effect of smoothing is that bagging reduces variance due to a soft-instead of a hard-thresholding operation.

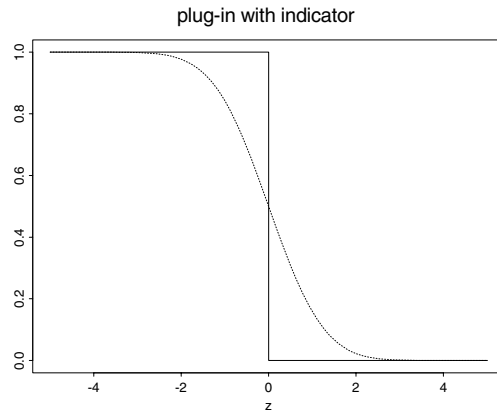


Figure 16.1. Indicator estimator from (16.3) at $x = x_n(0)$ as in (16.5). Function $L(z) = \mathbf{1}_{[z \leq 0]}$ (solid line) and $L_{\text{Bag}}(z)$ (dotted line) defining the asymptotics of the estimator in (16.6) and its bagged version in (16.7)

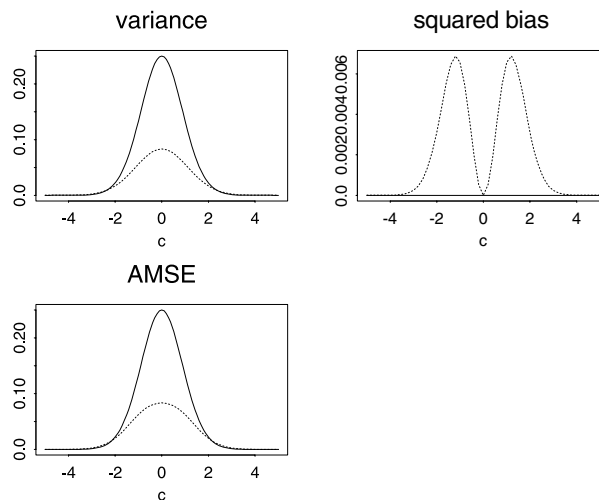


Figure 16.2. Indicator estimator from (16.3) at $x = x_n(c)$ as in (16.5). Asymptotic variance, squared bias and mean squared error (AMSE) (the target is $\lim_{n \rightarrow \infty} \mathbb{E}[\hat{g}(x)]$) for the estimator $\hat{g}(x_n(c))$ from (16.3) (solid line) and for the bagged estimator $\hat{g}_{\text{Bag}}(x_n(c))$ (dotted line) as a function of c

We can compute the first two asymptotic moments in the unstable region with $x = x_n(c)$. Numerical evaluations of these first two moments and the mean squared error (MSE) are given in Fig. 16.2. We see that in the approximate range where $|c| \leq 2.3$, bagging improves the asymptotic MSE. The biggest gain, by a factor 3, is at the most unstable point $x = \mu = \mathbb{E}[Y_1]$, corresponding to $c = 0$. The squared bias with bagging has only a negligible effect on the MSE (note the different scales in Fig. 16.2). Note that we always give an a-priori advantage to the original estimator which is asymptotically unbiased for the target as defined.

In (BY02), this kind of analysis has been given for more general estimators than \bar{Y}_n in (16.3) and also for estimation in linear models after testing. Hard decision indicator functions are involved there as well and bagging reduces variance due to its smoothing effect. The key to derive this property is always the fact that the bootstrap is asymptotically consistent as in (16.8).

Regression Trees

We address here the effect of bagging in the case of decision trees which are most often used in practice in conjunction with bagging. Decision trees consist of piecewise constant fitted functions whose supports (for the piecewise constants) are given by indicator functions similar to (16.3). Hence we expect bagging to bring a significant variance reduction as in the toy example above.

For simplicity of exposition, we consider first a one-dimensional predictor space and a so-called regression stump which is a regression tree with one split and two terminal nodes. The stump estimator (or algorithm) is then defined as the decision tree,

$$\hat{g}(x) = \hat{\beta}_\ell \mathbf{1}_{[x < \hat{d}]} + \hat{\beta}_u \mathbf{1}_{[x \geq \hat{d}]} = \hat{\beta}_\ell + (\hat{\beta}_u - \hat{\beta}_\ell) \mathbf{1}_{[\hat{d} \leq x]}, \tag{16.9}$$

where the estimates are obtained by least squares as

$$(\hat{\beta}_\ell, \hat{\beta}_u, \hat{d}) = \arg \min_{\beta_\ell, \beta_u, d} \sum_{i=1}^n (Y_i - \beta_\ell \mathbf{1}_{[X_i < d]} - \beta_u \mathbf{1}_{[X_i \geq d]})^2 .$$

These values are estimates for the best projected parameters defined by

$$(\beta_\ell^0, \beta_u^0, d^0) = \arg \min_{\beta_\ell, \beta_u, d} \mathbb{E} [(Y - \beta_\ell \mathbf{1}_{[X < d]} - \beta_u \mathbf{1}_{[X \geq d]})^2] . \tag{16.10}$$

The main mathematical difference of the stump in (16.9) to the toy estimator in (16.3) is the behavior of \hat{d} in comparison to the behavior of \bar{Y}_n (and not the constants $\hat{\beta}_\ell$ and $\hat{\beta}_u$ involved in the stump). It is shown in (BY02) that \hat{d} has convergence rate $n^{-1/3}$ (in case of a smooth regression function) and a limiting distribution which is non-Gaussian. This also explains that the bootstrap is not consistent, but consistency as in (16.8) turned out to be crucial in our analysis above. Bagging is still doing some kind of smoothing, but it is not known how this behaves quantitatively. However, a computationally attractive version of bagging, which has been found to perform often as good as bagging, turns out to be more tractable from a theoretical point of view.

16.2.3 Subagging

Subagging is a sobriquet for subsample aggregating where subsampling is used instead of the bootstrap for the aggregation. An estimator $\hat{g}(\cdot) = h_n((X_1, Y_1), \dots, (X_n, Y_n))(\cdot)$ is aggregated as follows:

$$\hat{g}_{SB(m)}(\cdot) = \binom{n}{m}^{-1} \sum_{(i_1, \dots, i_m) \in \mathcal{I}} h_m((X_{i_1}, Y_{i_1}), \dots, (X_{i_m}, Y_{i_m}))(\cdot),$$

where \mathcal{I} is the set of m -tuples ($m < n$) whose elements in $\{1, \dots, n\}$ are all distinct. This aggregation can be approximated by a stochastic computation. The subagging algorithm is as follows.

Subagging Algorithm

Step 1. For $k = 1, \dots, M$ (e.g. $M = 50$ or 100) do:

- (i) Generate a random subsample $(X_1^{*k}, Y_1^{*k}), \dots, (X_m^{*k}, Y_m^{*k})$ by randomly drawing m times without replacement from the data $(X_1, Y_1), \dots, (X_n, Y_n)$ (instead of resampling with replacement in bagging).
- (ii) Compute the subsampled estimator

$$\hat{g}_{(m)}^{*k}(\cdot) = h_m((X_1^{*k}, Y_1^{*k}), \dots, (X_m^{*k}, Y_m^{*k}))(\cdot);$$

Step 2. Average the subsampled estimators to approximate

$$\hat{g}_{SB(m)}(\cdot) \approx M^{-1} \sum_{k=1}^M \hat{g}_{(m)}^{*k}(\cdot).$$

As indicated in the notation, subagging depends on the subsample size m which is a tuning parameter (in contrast to M).

An interesting case is *half subagging* with $m = \lfloor n/2 \rfloor$. More generally, we could also use $m = \lfloor an \rfloor$ with $0 < a < 1$ (i.e. m a fraction of n) and we will argue why the usual choice $m = o(n)$ in subsampling for distribution estimation (PRW99) is a bad choice. Half subagging with $m = \lfloor n/2 \rfloor$ has been studied also in (BS02): in case where \hat{g} is a U -statistic, half subagging is exactly equivalent to bagging, and subagging yields very similar empirical results to bagging when the estimator $\hat{g}(\cdot)$ is a decision tree. Thus, if we don't want to optimize over the tuning parameter m , a good choice in practice is very often $m = \lfloor n/2 \rfloor$. Consequently, half subagging typically saves more than half of the computing time because the computational order of an estimator $\hat{g} = \hat{g}_{(n)}$ is usually at least linear in n .

Subagging Regression Trees

We describe here in a non-technical way the main mathematical result from (BY02) about subagging regression trees.

The underlying assumptions for some mathematical theory are as follows. The data generating regression model is

$$Y_i = g(X_i) + \varepsilon_i, \quad i = 1, \dots, n,$$

where X_1, \dots, X_n and $\varepsilon_1, \dots, \varepsilon_n$ are i.i.d. variables, independent from each other, and $\mathbb{E}[\varepsilon_1] = 0, \mathbb{E}|\varepsilon_1|^2 < \infty$. The regression function $g(\cdot)$ is assumed to be smooth and the distribution of X_i and ε_i are assumed to have suitably regular densities.

It is then shown in (BY02) that for $m = [an]$ ($0 < a < 1$),

$$\limsup_{n \rightarrow \infty} \frac{\mathbb{E}[(\hat{g}_{SB(m)}(x) - g(x))^2]}{\mathbb{E}[(\hat{g}_n(x) - g(x))^2]} < 1,$$

for x in suitable neighborhoods (depending on the fraction a) around the best projected split points of a regression tree (e.g. the parameter d^0 in (16.10) for a stump), and where $g(x) = \lim_{n \rightarrow \infty} \mathbb{E}[\hat{g}(x)]$. That is, subbagging asymptotically reduces the MSE for x in neighborhoods around the unstable split points, a fact which we may also compare with Fig. 16.2. Moreover, one can argue that globally,

$$\mathbb{E}[(\hat{g}_{SB(m)}(X) - g(X))^2] \stackrel{\text{approx.}}{<} \mathbb{E}[(\hat{g}(X) - g(X))^2]$$

for n large, and where the expectations are taken also over (new) predictors X .

For subbagging with small order $m = o(n)$, such a result is no longer true: the reason is that small order subbagging will then be dominated by a large bias (while variance reduction is even better than for fraction subbagging with $m = [an]$, $0 < a < 1$).

Similarly as for the toy example in Sect. 16.2.2, subbagging smoothes the hard decisions in a regression tree resulting in reduced variance and MSE.

Bagging More “Smooth” Base Procedures and Bragging 16.2.4

As discussed in Sects. 16.2.2 and 16.2.3, (su-)bagging smoothes out indicator functions which are inherent in some base procedures such as decision trees. For base procedures which are “smoother”, e.g. which do not involve hard decision indicators, the smoothing effect of bagging is expected to cause only small effects.

For example, in (BS02) it is proved that the effect of bagging on the MSE is only in the second order term if the base procedure is a U -statistic. Similarly, citing (CH03): “... when bagging is applied to relatively conventional statistical problems, it cannot reliably be expected to improve performance”. On the other hand, we routinely use nowadays “non-conventional” methods: a simple example is variable selection and fitting in a linear model where bagging has been demonstrated to improve predictive performance (Breg6a).

In (BDo2), the performance of bagging has been studied for MARS, projection pursuit regression and regression tree base procedures: most improvements of bagging are reported for decision trees. In (BY02), it is shown that bagging the basis function in MARS essentially doesn’t change the asymptotic MSE. In (Buho3) it is

empirically demonstrated in greater detail that for finite samples, bagging MARS is by far less effective – and sometimes very destructive – than bagging decision trees.

(Su-)bagging may also have a positive effect due to averaging over different selected predictor variables; this is an additional effect besides smoothing out indicator functions. In case of MARS, we could also envision that such an averaging over different selected predictor variables would have a positive effect: in the empirical analysis in (Buho3), this has been found to be only true when using a robust version of aggregation, see below.

16.2.5 Bragging

Bragging stands for bootstrap robust aggregating (Buho3): it uses the sample median over the M bootstrap estimates $\hat{g}^{*k}(\cdot)$, instead of the sample mean in Step 3 of the bagging algorithm.

While bragging regression trees was often found to be slightly less improving than bagging, bragging MARS seems better than the original MARS and much better than bagging MARS.

16.2.6 Out-of-Bag Error Estimation

Bagging “automatically” yields an estimate of the out-of-sample error, sometimes referred to as the generalization error. Consider a loss $\ell(Y, \hat{g}(X))$, measuring the discrepancy between an estimated function \hat{g} , evaluated at X , and the corresponding response Y , e.g. $\ell(Y, \hat{g}(X)) = |Y - \hat{g}(X)|^2$. The generalization error is then

$$err = \mathbb{E}[\ell(Y, \hat{g}(X))] ,$$

where the expectation \mathbb{E} is over the training data $(X_1, Y_1), \dots, (X_n, Y_n)$ (i.i.d. or stationary pairs), $\hat{g}(\cdot)$ a function of the training data, and (X, Y) is a new test observation, independent from the training data but having the same distribution as one training sample point (X_i, Y_i) .

In a bootstrap sample (in the bagging procedure), roughly $\exp(-1) \approx 37\%$ of the original observations are left out: they are called “out-of-bag” observations (Bre96b). Denote by $\mathcal{B}oot^k$ the original sample indices which were resampled in the k th bootstrap sample; note that the out-of-bag sample observations (in the k th bootstrap resampling stage) are then given by $\{1, \dots, n\} \setminus \mathcal{B}oot^k$ which can be used as test sets. The out-of-bag error estimate of bagging is then defined as

$$\widehat{err}_{OB} = n^{-1} \sum_{i=1}^n N_M^{-1} \sum_{k=1}^M \mathbf{1}_{[(X_i, Y_i) \notin \mathcal{B}oot^k]} \ell(Y_i, \hat{g}^{*k}(X_i)) ,$$

$$N_M = \sum_{k=1}^M \mathbf{1}_{[(X_i, Y_i) \notin \mathcal{B}oot^k]} .$$

In (Byl02), a correction of the out-of-bag error estimate is proposed. Out-of-bag estimation can also be used for other tasks, e.g. for more honest class probability estimates in classification when bagging trees (Breg6b).

Disadvantages

16.2.7

The main disadvantage of bagging, and other ensemble algorithms, is the lack of interpretation. A linear combination of decision trees is much harder to interpret than a single tree. Likewise: bagging a variable selection – fitting algorithm for linear models (e.g. selecting the variables using the AIC criterion within the least-squares estimation framework) gives little clues which of the predictor variables are actually important.

One way out of this lack of interpretation is sometimes given within the framework of bagging. In (ET98), the bootstrap has been justified to judge the importance of automatically selected variables by looking at relative appearance-frequencies in the bootstrap runs. The bagging estimator is the average of the fitted bootstrap functions, while the appearance frequencies of selected variables or interactions may serve for interpretation.

Other References

16.2.8

Bagging may also be useful as a “module” in other algorithms: BagBoosting (BY00) is a boosting algorithm (see Sect. 16.3) with a bagged base-procedure, often a bagged regression tree. The theory about bagging supports the finding that BagBoosting using bagged regression trees, which have smaller asymptotic MSEs than trees, is often better than boosting with regression trees. This is empirically demonstrated for a problem about tumor classification using microarray gene expression predictors (Deto4).

Bundling classifiers (HLo2), which is a more complicated aggregation algorithm but related to bagging, seems to perform better than bagging for classification. In (Rido2), bagging is used in conjunction with boosting (namely for stopping boosting iterations) for density estimation. In (DF03), bagging is used in the unsupervised context of cluster analysis, reporting improvements when using bagged clusters instead of original cluster-outputs.

Boosting

16.3

Boosting algorithms have been proposed in the machine learning literature by Schapire (Sch90) and Freund (Fre95, FS96), see also (Scho2). These first algorithms have been developed as ensemble methods. Unlike bagging which is a parallel ensemble method, boosting methods are sequential ensemble algorithms where the weights c_k in (16.1) are depending on the previous fitted functions $\hat{g}_1, \dots, \hat{g}_{k-1}$.

Boosting has been empirically demonstrated to be very accurate in terms of classification, notably the so-called AdaBoost algorithm (FS96).

We will explain below that boosting can be viewed as a nonparametric optimization algorithm in function space, as first pointed out by Breiman (Bre98, Bre99a). This view turns out to be very fruitful to adapt boosting for other problems than classification, including regression and survival analysis.

Maybe it is worth mentioning here that boosting algorithms have often better predictive power than bagging, (cf. (Bre98)); of course, such a statement has to be read with caution, and methods should be tried out on individual data-sets, including e.g. cross-validation, before selecting one among a few methods.

To give an idea, we report here some empirical results from (Bre98) for classification: we show below the gains (in percentage) of boosting trees over bagging trees:

“normal” size data-sets: 64.3%, 10.8%, 20.3%, -4.6%, 6.9%, 16.2% ,

large data-sets: 37.5%, 12.6%, -50.0%, 4.0%, 28.6% .

For all data-sets, boosting trees was better than a single classification tree. The biggest loss of 50% for boosting in comparison with bagging is for a data-set with very low misclassification error, where bagging achieves 0.014% and boosting 0.021%.

16.3.1 Boosting as Functional Gradient Descent

Rather than looking through the lenses of ensemble methods, boosting algorithms can be seen as functional gradient descent techniques (Bre98, Bre99a). The goal is to estimate a function $g : \mathbb{R}^d \rightarrow \mathbb{R}$, minimizing an expected loss

$$\mathbb{E}[\ell(Y, g(X))] , \ell(\cdot, \cdot) : \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}^+ , \quad (16.11)$$

based on data (X_i, Y_i) ($i = 1, \dots, n$) as in Sect. 16.2.1. The loss function ℓ is typically assumed to be convex in the second argument. We consider here both cases where the univariate response Y is continuous (regression problem) or discrete (classification problem), since boosting is potentially useful in both cases.

As we will see in Sect. 16.3.2, boosting algorithms are pursuing a “small” empirical risk

$$n^{-1} \sum_{i=1}^n \ell(Y_i, g(X_i))$$

by selecting a g in the linear hull of some function class, i.e. $g(\cdot) = \sum_k c_k g_k(\cdot)$ with $g_k(\cdot)$'s from a function class such as trees.

The most popular loss functions, for regression and binary classification, are given in Table 16.1.

Table 16.1. The squared error, binomial negative log-likelihood and exponential loss functions and their population minimizers; $\text{logit}(p) = \log(p/(1 - p))$

Boosting	Loss function	Population minimizer for (16.11)
L_2 Boost	$\ell(y, g) = (y - g)^2$	$g(x) = \mathbb{E}[Y X = x]$
LogitBoost	$\ell(y, g) = \log_2(1 + \exp(-2(y - 1)g))$	$g(x) = 0.5 \cdot \text{logit}(\mathbb{P}[Y = 1 X = x])$
AdaBoost	$\ell(y, g) = \exp(-(2y - 1)g)$	$g(x) = 0.5 \cdot \text{logit}(\mathbb{P}[Y = 1 X = x])$

While the squared error loss is mainly used for regression (see (BY03) for classification with the squared error loss), the log-likelihood and the exponential loss are for binary classification only.

The Margin for Classification

The form of the log-likelihood loss may be somewhat unusual: we norm it, by using the base 2 so that it “touches” the misclassification error as an upper bound (see Fig. 16.3), and we write it as a function of the so-called *margin* $\tilde{y}g$, where $\tilde{y} = 2y - 1 \in \{-1, 1\}$ is the usual labeling from the machine learning community. Thus, the loss is a function of the margin $\tilde{y}g$ only; and the same is true with the exponential loss and also the squared error loss for classification since

$$(\tilde{y} - g)^2 = \tilde{y}^2 - 2\tilde{y}g + g^2 = 1 - 2\tilde{y}g + (\tilde{y}g)^2,$$

using $\tilde{y}^2 = 1$.

The misclassification loss, or zero-one loss, is $\mathbf{1}_{[\tilde{y}g < 0]}$, again a function of the margin, whose population minimizer is $g(x) = \mathbf{1}_{[\mathbb{P}[Y=1|X=x] > 1/2]}$. For readers less

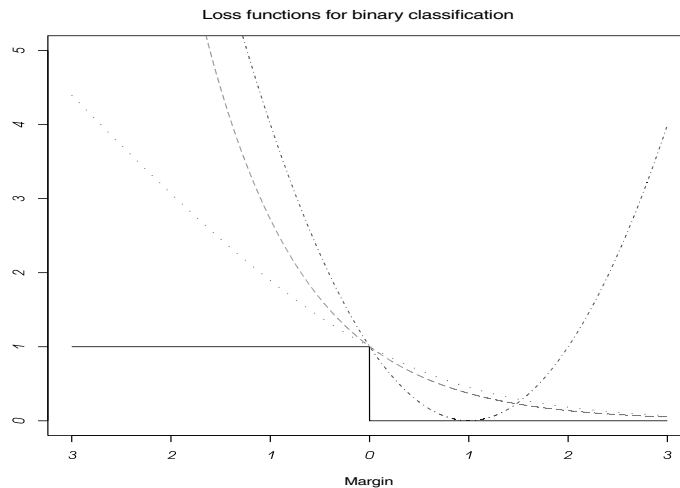


Figure 16.3. Loss functions of the margin for binary classification. Zero-one misclassification loss (solid line), log-likelihood loss (dashed line), exponential loss (dotted line), squared error loss (dashed/dotted). The loss-functions are described in Table 16.1

familiar with the concept of the margin, this can also be understood as follows: the Bayes-classifier which minimizes the misclassification risk is

$$g_{\text{Bayes}}(x) = \mathbf{1}_{[\mathbb{P}[Y=1|X=x]>1/2]} .$$

We can now see that a misclassification occurs, if $y = 0$, $g_{\text{Bayes}}(x) = 1$ or $y = 1$, $g_{\text{Bayes}}(x) = 0$, which is equivalent to $2(y - 1)g_{\text{Bayes}}(x) < 0$ or $\tilde{y}g_{\text{Bayes}}(x) < 0$.

The (surrogate) loss functions given in Table 16.1 are all convex functions of the margin $\tilde{y}g$ which bound the zero-one misclassification loss from above, see Fig. 16.3. The convexity of these surrogate loss functions is computationally important for empirical risk minimization; minimizing the empirical zero-one loss is computationally intractable.

16.3.2 The Generic Boosting Algorithm

Estimation of the function $g(\cdot)$, which minimizes an expected loss in (16.11), is pursued by a constrained minimization of the empirical risk $n^{-1} \sum_{i=1}^n \ell(Y_i, g(X_i))$. The constraint comes in algorithmically (and not explicitly), by the way we are attempting to minimize the empirical risk, with a so-called functional gradient descent. This gradient descent view has been recognized and refined by various authors (cf. (Bre98, Bre99a, MBBFoo, FHToo, Frio1, BY03)). In summary, the minimizer of the empirical risk is imposed to satisfy a “smoothness” constraint in terms of a linear expansion of (“simple”) fits from a real-valued base procedure function estimate.

Generic Functional Gradient Descent

Step 1 (initialization). Given data $\{(X_i, Y_i); i = 1, \dots, n\}$, apply the base procedure yielding the function estimate

$$\hat{F}_1(\cdot) = \hat{g}(\cdot) ,$$

where $\hat{g} = \hat{g}_{X,Y} = h_n((X_1, Y_1), \dots, (X_n, Y_n))$ is a function of the original data. Set $m = 1$.

Step 2 (projecting gradient to learner). Compute the negative gradient vector

$$U_i = - \frac{\partial \ell(Y_i, g)}{\partial g} \Big|_{g=\hat{F}_m(X_i)} , \quad i = 1, \dots, n ,$$

evaluated at the current $\hat{F}_m(\cdot)$. Then, apply the base procedure to the gradient vector

$$\hat{g}_{m+1}(\cdot) ,$$

where $\hat{g}_{m+1} = \hat{g}_{X,U} = h_n((X_1, U_1), \dots, (X_n, U_n))$ is a function of the original predictor variables and the current negative gradient vector as pseudo-response.

Step 3 (line search). Do a one-dimensional numerical search for the best step-size

$$\hat{s}_{m+1} = \arg \min_s \sum_{i=1}^n \ell(Y_i, \hat{F}_m(X_i) + s\hat{g}_{m+1}(X_i)) .$$

Update,

$$\hat{F}_{m+1}(\cdot) = \hat{F}_m(\cdot) + \hat{s}_{m+1}\hat{g}_{m+1}(\cdot) .$$

Step 4 (iteration). Increase m by one and repeat Steps 2 and 3 until a stopping iteration M is achieved.

The number of iterations M is the tuning parameter of boosting. The larger it is, the more complex the estimator. But the complexity, for example the variance of the estimator, is not linearly increasing in M : instead, it increases very slowly as M gets larger, see also Fig. 16.4 in Sect. 16.3.5.

Obviously, the choice of the base procedure influences the boosting estimate. Originally, boosting has been mainly used with tree-type base procedures, typically with small trees such as stumps (two terminal nodes) or trees having say 8 terminal nodes (cf. (Bre98, Bre04, BK99, FHT00, DB03)); see also Sect. 16.3.8. But we will demonstrate in Sect. 16.3.6 that boosting may be very worthwhile within the class of linear, additive or interaction models, allowing for good model interpretation.

The function estimate \hat{g}_{m+1} in Step 2 can be viewed as an estimate of $\mathbb{E}[U_i|X = x]$, the expected negative gradient given the predictor X , and takes values in \mathbb{R} , even in case of a classification problem with Y_i in a finite set (this is different from the AdaBoost algorithm, see below).

We call $\hat{F}_M(\cdot)$ the L_2 Boost-, LogitBoost- or AdaBoost-estimate, according to the implementing loss function $(y - g)^2$, $\log_2(1 + \exp(-2(y - 1)g))$ or $\ell(y, g) = \exp(-(2y - 1)g)$, respectively; see Table 16.1.

The original AdaBoost algorithm for classification is actually a bit different: the base procedure fit is a classifier, and not a real-valued estimator for the conditional probability of Y given X ; and Steps 2 and 3 are also somewhat different. Since AdaBoost's implementing exponential loss function is not well established in statistics, we refer for a detailed discussion to (FHT00). From a statistical perspective, the squared error loss and log-likelihood loss functions are most prominent and we describe below the corresponding boosting algorithms in detail.

L_2 Boost

Boosting using the squared error loss, L_2 Boost, has a simple structure: the negative gradient in Step 2 is the classical residual vector and the line search in Step 3 is trivial when using a base procedure which does least squares fitting.

L_2 Boost Algorithm

Step 1 (initialization). As in Step 1 of generic functional gradient descent.

Step 2. Compute residuals $U_i = Y_i - \hat{F}_m(X_i)$ ($i = 1, \dots, n$) and fit the real-valued base procedure to the current residuals (typically by (penalized) least squares) as in Step 2 of the generic functional gradient descent; the fit is denoted by $\hat{g}_{m+1}(\cdot)$.

Update

$$\hat{F}_{m+1}(\cdot) = \hat{F}_m(\cdot) + \hat{g}_{m+1}(\cdot).$$

We remark here that, assuming the base procedure does some (potentially penalized) least squares fitting of the residuals, the line search in Step 3 of the generic algorithm becomes trivial with $\hat{s}_{m+1} = 1$.

Step 3 (iteration). Increase iteration index m by one and repeat Step 2 until a stopping iteration M is achieved.

The estimate $\hat{F}_M(\cdot)$ is an estimator of the regression function $\mathbb{E}[Y|X = \cdot]$. L_2 Boosting is nothing else than repeated least squares fitting of residuals (cf. (Frio1, BY03)). With $m = 2$ (one boosting step), it has already been proposed by Tukey (Tuk77) under the name “twicing”. In the non-stochastic context, the L_2 Boosting algorithm is known as “Matching Pursuit” (MZ93) which is popular in signal processing for fitting overcomplete dictionaries.

LogitBoost

Boosting using the log-likelihood loss for binary classification (and more generally for multi-class problems) is known as LogitBoost (FHT00). LogitBoost uses some Newton-stepping with the Hessian, rather than the line search in Step 3 of the generic boosting algorithm:

LogitBoost Algorithm

Step 1 (initialization). Start with conditional probability estimates $\hat{p}_1(X_i) = 1/2$ ($i = 1, \dots, n$) (for $\mathbb{P}[Y = 1|X = X_i]$). Set $m = 1$.

Step 2. Compute the pseudo-response (negative gradient)

$$U_i = \frac{Y_i - \hat{p}_m(X_i)}{\hat{p}_m(X_i)(1 - \hat{p}_m(X_i))},$$

and the weights

$$w_i = \hat{p}_m(X_i) (1 - \hat{p}_m(X_i)).$$

Fit the real-valued base procedure to the current pseudo-response U_i ($i = 1, \dots, n$) by weighted least squares, using the current weights w_i ($i = 1, \dots, n$); the fit is denoted by $\hat{g}_{m+1}(\cdot)$. Update

$$\hat{F}_{m+1}(\cdot) = \hat{F}_m(\cdot) + 0.5 \cdot \hat{g}_{m+1}(\cdot)$$

and

$$\hat{p}_{m+1}(X_i) = \frac{\exp(\hat{F}_{m+1}(X_i))}{\exp(\hat{F}_{m+1}(X_i)) + \exp(-\hat{F}_{m+1}(X_i))}.$$

Step 3 (iteration). Increase iteration index m by one and repeat Step 2 until a stopping iteration M is achieved.

The estimate $\hat{F}_M(\cdot)$ is an estimator for half of the log-odds ratio $0.5 \cdot \text{logit}(\mathbb{P}[Y = 1|X = \cdot])$ (see Table 16.1). Thus, a classifier (under equal misclassification loss for the labels $Y = 0$ and $Y = 1$) is

$$\text{sign}(\hat{F}_M(\cdot)),$$

and an estimate for the conditional probability $\mathbb{P}[Y = 1|X = \cdot]$ is

$$\hat{p}_M(\cdot) = \frac{\exp(\hat{F}_M(\cdot))}{\exp(\hat{F}_M(\cdot)) + \exp(-\hat{F}_M(\cdot))}.$$

A requirement for LogitBoost is that the base procedure has the option to be fitted by *weighted* least squares.

Multi-class Problems

The LogitBoost algorithm described above can be modified for multi-class problems where the response variable takes values in a finite set $\{0, 1, \dots, J - 1\}$ with $J > 2$ by using the multinomial log-likelihood loss (FHT00). But sometimes it can be advantageous to run instead a binary classifier (e.g. with boosting) for many binary problems. The most common approach is to code for J binary problems where the j th problem assigns the response

$$Y^{(j)} = \begin{cases} 1, & \text{if } Y = j, \\ 0, & \text{if } Y \neq j. \end{cases}$$

i.e. the so-called “one versus all” approach. For example, if single class-label can be distinguished well from all others, the “one versus all” approach seems adequate: empirically, this has been reported for classifying tumor types based on microarray gene expressions when using a LogitBoost algorithm (DB03).

Other codings of a multi-class into multiple binary problems are discussed in (ASS01).

16.3.3 Small Step Size

It is often better to use small step sizes instead of using the full line search step-length \hat{s}_{m+1} from Step 3 in the generic boosting algorithm (or $\hat{s}_{m+1} \equiv 1$ for L_2 Boost or $\hat{s}_{m+1} \equiv 0.5$ for LogitBoost). We advocate here to use the step-size

$$\nu \hat{s}_{m+1}, \quad 0 < \nu \leq 1,$$

where ν is constant during boosting iterations and small, e.g. $\nu = 0.1$. The parameter ν can be seen as a simple shrinkage parameter, where we use the shrunken $\nu \hat{g}_{m+1}(\cdot)$ instead of the unshrunken $\hat{g}_{m+1}(\cdot)$. Small step-sizes (or shrinkage) make the boosting algorithm slower and require a larger number M of iterations. However, the computational slow-down often turns out to be advantageous for better out-of-sample prediction performance, (cf. (Frio1, BY03)). There are also some theoretical reasons to use boosting with ν (infinitesimally) small (EHJT03).

16.3.4 The Bias-variance Trade-off for L_2 Boost

We discuss here the behavior of boosting in terms of model-complexity and estimation error when the number of iterations increase. This is best understood in the framework of squared error loss and L_2 Boosting.

We represent the base procedure as an operator

$$\mathcal{J} : \mathbb{R}^n \rightarrow \mathbb{R}^n, \quad (U_1, \dots, U_n)^T \mapsto (\hat{U}_1, \dots, \hat{U}_n)^T$$

which maps a (pseudo-)response vector $(U_1, \dots, U_n)^T$ to its fitted values; the predictor variables X are absorbed here into the operator notation. That is,

$$\mathcal{J}(U_1, \dots, U_n)^T = (\hat{g}(X_1), \dots, \hat{g}(X_n))^T,$$

where $\hat{g}(\cdot) = \hat{g}_{X,U}(\cdot)$ is the estimate from the base procedure based on data (X_i, U_i) , $i = 1, \dots, n$. Then, the boosting operator in iteration m equals

$$\mathcal{B}_m = I - (I - \mathcal{J})^m$$

and the fitted values of boosting after m iterations are

$$\mathcal{B}_m Y = Y - (I - \mathcal{J})^m Y, \quad Y = (Y_1, \dots, Y_n)^T.$$

Heuristically, if the base procedure satisfies $\|I - \mathcal{J}\| < 1$ for a suitable norm, i.e. has a “learning capacity” such that the residual vector is shorter than the input-response vector, we see that \mathcal{B}_m converges to the identity I as $m \rightarrow \infty$, and $\mathcal{B}_m Y$ converges to the fully saturated model Y as $m \rightarrow \infty$, interpolating the response data exactly. Thus, we have to stop the boosting algorithm at some suitable iteration number $m = M$, and we see that a bias-variance trade-off is involved when varying the iteration number M .

L_2 Boost with Smoothing Spline Base Procedure for One-dimensional Curve Estimation

16.3.5

The case where the base procedure is a smoothing spline for a one-dimensional predictor $X \in \mathbb{R}^1$ is instructive, although being only a toy example within the range of potential applications of boosting algorithms.

In our notation from above, \mathcal{S} denotes a smoothing spline operator which is the solution ($\mathcal{S}Y = g(X_1), \dots, f(X_n)$) of the following optimization problem (cf. (Wah90))

$$\arg \min_g n^{-1} \sum_{i=1}^n (Y_i - g(X_i))^2 + \lambda \int g''(x)^2 dx .$$

The smoothing parameter λ controls the bias-variance trade-off, and tuning the smoothing spline estimator usually boils down to estimating a good value of λ . Alternatively, the L_2 Boosting approach for curve-estimation with a smoothing spline base procedure is as follows.

Choosing the Base Procedure

Within the class of smoothing spline base procedures, we choose a spline by fixing a smoothing parameter λ . This should be done such that the base procedure has low variance but potentially high bias: for example, we may choose λ such that the degrees of freedom $df = \text{trace}(\mathcal{S})$ is low, e.g. $df = 2.5$. Although the base procedure has typically high bias, we will reduce it by pursuing suitably many boosting iterations. Choosing the df is not really a tuning parameter: we only have to make sure that df is small enough, so that the initial estimate (or first few boosting estimates) are not already overfitting. This is easy to achieve in practice and a theoretical characterization is described in (BY03).

Related aspects of choosing the base procedure are described in Sects. 16.3.6 and 16.3.8. The general “principle” is to choose a base procedure which has low variance and having the property that when taking linear combinations thereof, we obtain a model-class which is rich enough for the application at hand.

MSE Trace and Stopping

As boosting iterations proceed, the bias of the estimator will go down and the variance will increase. However, this bias-variance exhibits a very different behavior as when classically varying the smoothing parameter (the parameter λ). It can be shown that the variance increases with exponentially small increments of the order $\exp(-Cm)$, $C > 0$, while the bias decays quickly: the optimal mean squared error for the best boosting iteration m is (essentially) the same as for the optimally selected tuning parameter λ (BY03), but the trace of the mean squared error is very different, see Fig. 16.4. The L_2 Boosting method is much less sensitive to overfitting and hence often easier to tune. The mentioned insensitivity about overfitting also applies to higher-dimensional problems, implying potential advantages about tuning.

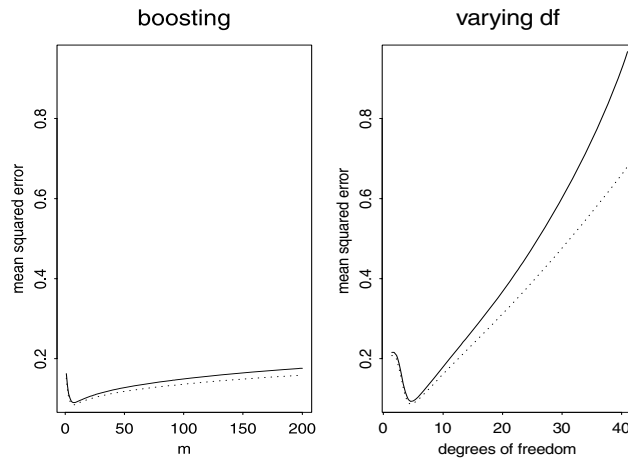


Figure 16.4. Mean squared error $\mathbb{E}[(g(X) - \hat{g}(X))^2]$ for new predictor X (solid line) and $n^{-1} \sum_{i=1}^n \mathbb{E}[(\hat{F}_m(X_i) - g(X_i))^2]$ (dotted line) from 100 simulations of a nonparametric regression model with smooth regression function and $\text{Unif.}[-1/2, 1/2]$ -distributed design points. Sample size is $n = 100$. *Left:* L_2 Boost with cubic smoothing spline having $df = 3$, as a function of boosting iterations m . *Right:* Cubic smoothing spline for various degrees of freedom (various amount of smoothing)

Asymptotic Optimality

Such L_2 Boosting with smoothing splines achieves the asymptotically optimal min-max MSE rates, and the method can even adapt to higher order smoothness of the true underlying function, without knowledge of the true degree of smoothness (BY03).

16.3.6 L_2 Boost for Additive and Interaction Regression Models

In Sect. 16.3.4, we already pointed out that L_2 Boosting yields another way of regularization by seeking for a compromise between bias and variance. This regularization turns out to be particularly powerful in the context with many predictor variables.

Additive Modeling

Consider the component-wise smoothing spline which is defined as a smoothing spline with *one selected* predictor variable $X^{(i)}$ ($i \in \{1, \dots, d\}$), where

$$\hat{i} = \arg \min_i \sum_{i=1}^n (Y_i - \hat{g}_i(X_i^{(i)}))^2,$$

and \hat{g}_i are smoothing splines with single predictors $X^{(j)}$, all having the same low degrees of freedom df , e.g. $df = 2.5$.

L_2 Boost with component-wise smoothing splines yields an additive model, since in every boosting iteration, a function of one selected predictor variable is

linearly added to the current fit and hence, we can always rearrange the summands to represent the boosting estimator as an additive function in the original variables, $\sum_{j=1}^d \hat{m}_j(x_j)$, $x \in \mathbb{R}^d$. The estimated functions $\hat{m}_j(\cdot)$ are fitted in a stage-wise fashion and they are different from the backfitting estimates in additive models (cf. (HT90)). Boosting has much greater flexibility to add complexity, in a stage-wise fashion: in particular, boosting does variable selection, since some of the predictors will never be chosen, and it assigns variable amount of degrees of freedom to the selected components (or function estimates); the degrees of freedom are defined below. An illustration of this interesting way to fit additive regression models with high-dimensional predictors is given in Figs. 16.5 and 16.6 (actually, a penalized version of L_2 Boosting, as described below, is shown).

When using regression stumps (decision trees having two terminal nodes) as the base procedure, we also get an additive model fit (by the same argument as with component-wise smoothing splines). If the additive terms $m_j(\cdot)$ are smooth functions of the predictor variables, the component-wise smoothing spline is often a better base procedure than stumps (BY03). For the purpose of classification, e.g. with LogitBoost, stumps often seem to do a decent job; also, if the predictor variables are non-continuous, component-wise smoothing splines are often inadequate.

Finally, if the number d of predictors is “reasonable” in relation to sample size n , boosting techniques are not necessarily better than more classical estimation methods (BY03). It seems that boosting has most potential when the predictor dimension is very high (BY03). Presumably, more classical methods become then very difficult to tune while boosting seems to produce a set of solutions (for every boosting iteration another solution) whose best member, chosen e.g. via cross-validation, has often very good predictive performance. A reason for the efficiency of the trace of boosting solutions is given in Sect. 16.3.9.

Degrees of Freedom and AIC_c -stopping Estimates

For component-wise base procedures, which pick one or also a pair of variables at the time, all the component-wise fitting operators are involved: for simplicity, we focus on additive modeling with component-wise fitting operators \mathcal{S}_j , $j = 1, \dots, d$, e.g. the component-wise smoothing spline.

The boosting operator, when using the step size $0 < \nu \leq 1$, is then of the form

$$\mathcal{B}_m = I - (I - \nu \mathcal{S}_{i_1}) (I - \nu \mathcal{S}_{i_2}) \dots (I - \nu \mathcal{S}_{i_m}) ,$$

where $i_i \in \{1, \dots, d\}$ denotes the component which is picked in the component-wise smoothing spline in the i th boosting iteration.

If the \mathcal{S}_j 's are all linear operators, and ignoring the effect of selecting the components, it is reasonable to define the degrees of boosting as

$$df(\mathcal{B}_m) = \text{trace}(\mathcal{B}_m) .$$

We can represent

$$\mathcal{B}_m = \sum_{j=1}^d M_j,$$

where $M_j = M_{j,m}$ is the linear operator which yields the fitted values for the j th additive term, e.g. $M_j \mathbf{Y} = (\hat{m}_j(X_1), \dots, \hat{m}_j(X_n))^T$. Note that the M_j 's can be easily computed in an iterative way by up-dating in the i th boosting iteration as follows:

$$M_{i,\text{new}} \leftarrow M_{i,\text{old}} + \nu \delta_i (I - \mathcal{B}_{i-1})$$

and all other M_j , $j \neq i$ do not change. Thus, we have a decomposition of the total degrees of freedom into the d additive terms:

$$df(\mathcal{B}_m) = \sum_{j=1}^d df_{j,m},$$

$$df_{j,m} = \text{trace}(M_j).$$

The individual degrees of freedom $df_{j,m}$ are a useful measure to quantify the complexity of the j th additive function estimate $\hat{m}_j(\cdot)$ in boosting iteration m . Note that $df_{j,m}$ will increase very sub-linearly as a function of boosting iterations m , see also Fig. 16.4.

Having some degrees of freedom at hand, we can now use the AIC, or some corrected version thereof, to define a stopping rule of boosting without doing some sort of cross-validation: the corrected AIC statistic (HST98) for boosting in the m th iteration is

$$AIC_c = \log(\hat{\sigma}^2) + \frac{1 + \text{trace}(\mathcal{B}_m)/n}{1 - (\text{trace}(\mathcal{B}_m) + 2)/n}, \quad (16.12)$$

$$\hat{\sigma}^2 = n^{-1} \sum_{i=1}^n (Y_i - (\mathcal{B}_m \mathbf{Y})_i)^2. \quad (16.13)$$

Alternatively, we could use generalized cross-validation (cf. (HTF01)), which involves degrees of freedom. This would exhibit the same computational advantage, as AIC_c , over cross-validation: instead of running boosting multiple times, AIC_c and generalized cross-validation need only one run of boosting (over a suitable number of iterations).

Penalized L_2 Boosting

When viewing the AIC_c criterion in (16.12) as a reasonable estimate of the true underlying mean squared error (ignoring uninteresting constants), we may attempt to construct a boosting algorithm which reduces in every step the AIC_c statistic (an estimate of the out-sample MSE) most, instead of maximally reducing the in-sample residual sum of squares.

We describe here penalized boosting for additive model fitting using individual smoothing splines:

Penalized L_2 Boost with Additive Smoothing Splines

Step 1 (initialization). As in Step 1 of L_2 Boost by fitting a component-wise smoothing spline.

Step 2. Compute residuals $U_i = Y_i - \hat{F}_m(X_i)$ ($i = 1, \dots, n$). Choose the individual smoothing spline which reduces AIC_c most: denote the selected component by \hat{i}_{m+1} and the fitted function, using the selected component \hat{i}_{m+1} by $\hat{g}_{m+1}(\cdot)$.

Update

$$\hat{F}_{m+1}(\cdot) = \hat{F}_m(\cdot) + \nu \hat{g}_{m+1}(\cdot).$$

for some step size $0 < \nu \leq 1$.

Step 3 (iteration). Increase iteration index m by one and repeat Step 2 until the AIC_c criterion in (16.12) cannot be improved anymore.

This algorithm cannot be written in terms of fitting a base procedure multiple times since selecting the component i in Step 2 not only depends on the residuals U_1, \dots, U_n , but also on the degrees of boosting, i.e. $\text{trace}(\mathcal{B}_{m+1})$; the latter is a complicated, although linear function, of the boosting iterations $m' \in \{1, 2, \dots, m\}$. Penalized L_2 Boost yields more sparse solutions than the corresponding L_2 Boost (with component-wise smoothing splines as corresponding base procedure). The reason is that $df_{j,m}$ increases only little in iteration $m + 1$, if the j th selected predictor variables has already been selected many times in previous iterations; this is directly connected to the slow increase in variance and overfitting as exemplified in Fig. 16.4.

An illustration of penalized L_2 Boosting with individual smoothing splines is shown in Figs. 16.5 and 16.6, based on simulated data. The simulation model is

$$\begin{aligned} X_1, \dots, X_n \text{ i.i.d. } &\sim \text{Unif.}[0, 1]^{100}, \\ Y_i &= \sum_{j=1}^{10} m_j(X^{(j)}) + \varepsilon_i \quad (i = 1, \dots, n), \\ \varepsilon_1, \dots, \varepsilon_n \text{ i.i.d. } &\sim \mathcal{N}(0, 0.5), \end{aligned} \tag{16.14}$$

where the m_j 's are smooth curves having varying curve complexities, as illustrated in Fig. 16.6. Sample size is $n = 200$ which is small in comparison to $d = 100$ (but the effective number of predictors is only 10).

In terms of prediction performance, penalized L_2 Boosting is not always better than L_2 Boosting; Fig. 16.7 illustrates an advantage of penalized L_2 Boosting.

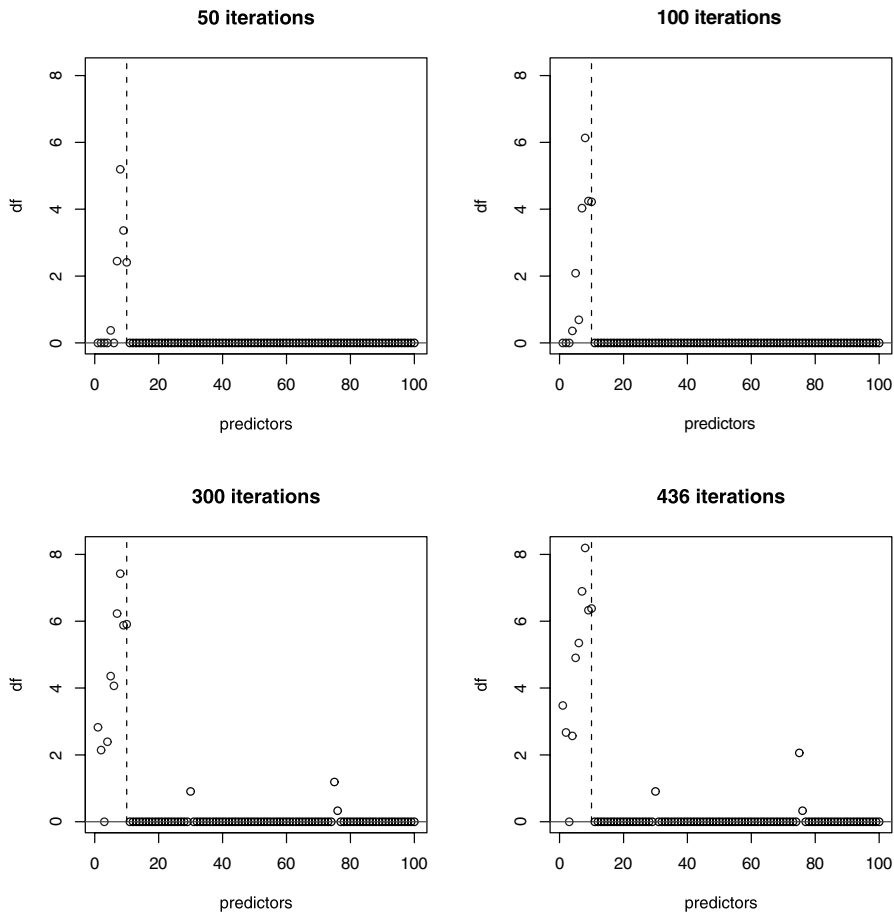


Figure 16.5. Degrees of freedom (df) in additive model fitting for all 100 predictor variables (from model (16.14)) during the process of penalized L_2 Boosting with individual smoothing splines (having $df = \text{trace}(\mathcal{S}_j) = 2.5$ for each spline). The first ten predictor variables (separated by the dashed line) are effective. The result is based on one realization from model (16.14) with sample size $n = 200$. The plot on the lower right corresponds to the estimated optimal number of boosting iterations using the AIC_c criterion in (16.12). Only three non-effective predictors have been selected (and assigned small amount of df), and one effective predictor has not been selected (but whose true underlying function is close to the zero-line, see Fig. 16.6)

But penalized L_2 Boosting is always sparser (or at least not less sparse) than the corresponding L_2 Boosting.

Obviously, penalized L_2 Boosting can be used for other than additive smoothing spline model fitting. The modifications are straightforward as long as the individual base procedures are linear operators.

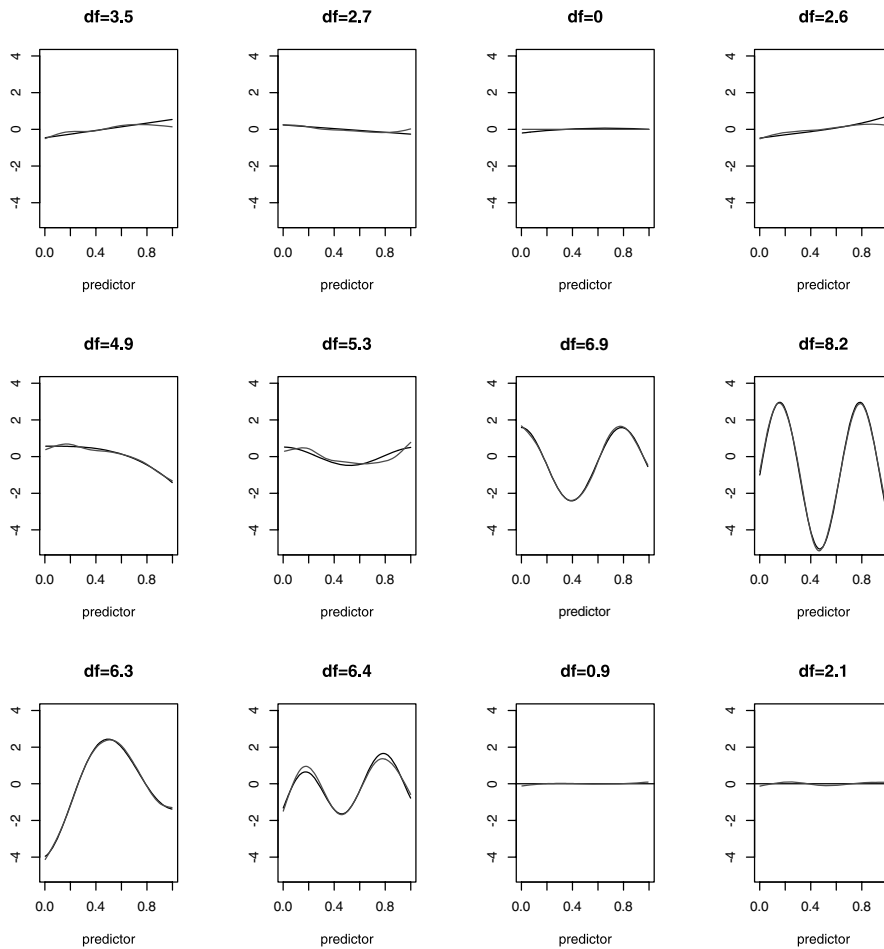


Figure 16.6. True underlying additive regression curves (*dashed lines*) and estimates (*solid lines*) from penalized L_2 Boosting as described in Fig. 16.5 (using 436 iterations, estimated from (16.12)). The last two plots correspond to non-effective predictors (the true functions are the zero-line), where L_2 Boosting assigned most df among non-effective predictors

Interaction Modeling

L_2 Boosting for additive modeling can be easily extended to interaction modeling (having low degree of interaction). Among the most prominent case is the second order interaction model $\sum_{j,k=1}^d \hat{m}_{j,k}(x_j, x_k)$, where $\hat{m}_{j,k} : \mathbb{R}^2 \rightarrow \mathbb{R}$.

Boosting with a pairwise thin plate spline, which selects the best pair of predictor variables yielding lowest residual sum of squares (when having the same degrees of freedom for every thin plate spline), yields a second-order interaction model. We demonstrate in Fig. 16.7 the effectiveness of this procedure in comparison with the second-order MARS fit (Fri91). The underlying model is the Friedman #1 model:

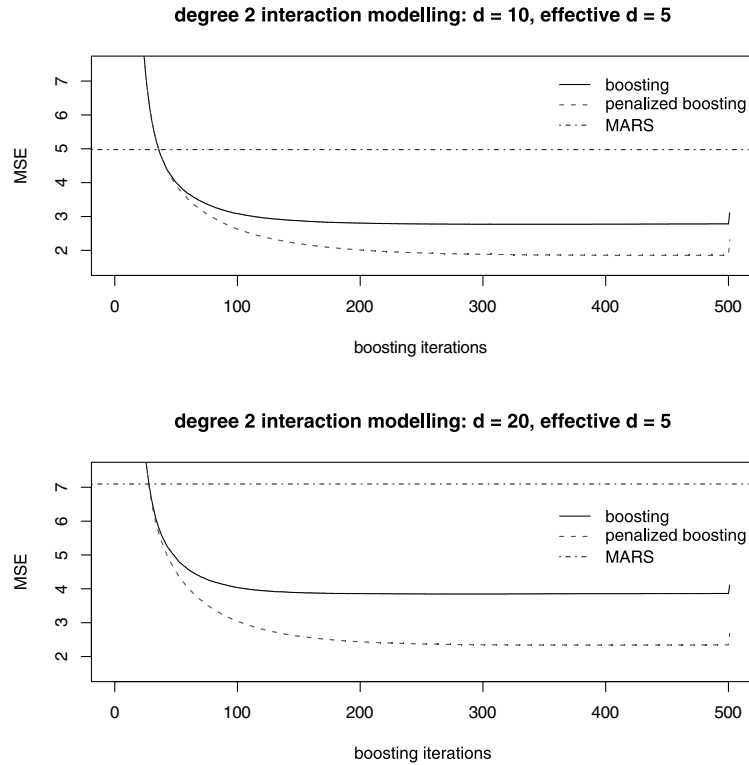


Figure 16.7. Mean squared errors for L_2 Boost with pairwise thin-plate splines (of two predictor variables, having $df = \text{trace}(\mathcal{B}_{i,k}) = 2.5$) (solid lines), its penalized version (dashed lines) and MARS restricted to the (correct) second order interactions (dashed/dotted lines). The point with abscissa $x = 501$ for the boosting methods corresponds to the performance when estimating the number of iterations using (16.12). Based on simulated data from model (16.15) with $n = 50$

$$\begin{aligned}
 &X_1, \dots, X_n \text{ i.i.d. } \sim \text{Unif.}([0, 1]^d), \quad d \in \{10, 20\}, \\
 &Y_i = 10 \sin(\pi X^{(1)} X^{(2)}) + 20 (X^{(3)} - 0.5)^2 + 10X^{(4)} + 5X^{(5)} + \varepsilon_i \\
 &\quad (i = 1, \dots, n), \\
 &\varepsilon_1, \dots, \varepsilon_n \text{ i.i.d. } \sim \mathcal{N}(0, 1). \tag{16.15}
 \end{aligned}$$

The sample size is chosen as $n = 50$ which is small in comparison to $d = 20$.

In high-dimensional settings, it seems that such interaction L_2 Boosting is clearly better than the more classical MARS fit, while both of them share the same superb simplicity of interpretation.

Linear Modeling

16.3.7

L_2 Boosting turns out to be also very useful for linear models when there are many predictor variables. An attractive base procedure is component-wise linear least squares regression, using the one selected predictor variables which reduces residual sum of squares most.

This method does variable selection, since some of the predictors will never be picked during boosting iterations; and it assigns variable amount of degrees of freedom (or shrinkage), as discussed for additive models above. Recent theory shows that this method is consistent for very high-dimensional problems where the number of predictors $d = d_n$ is allowed to grow like $\exp(Cn)$ ($C > 0$), but the true underlying regression coefficients are sparse in terms of their ℓ_1 -norm, i.e. $\sup_n \|\beta\|_1 = \sup_n \sum_{j=1}^{d_n} |\beta_j| < \infty$, where β is the vector of regression coefficients (Buh04).

Boosting Trees

16.3.8

The most popular base procedures for boosting, at least in the machine learning community, are trees. This may be adequate for classification, but when it comes to regression, or also estimation of conditional probabilities $\mathbb{P}[Y = 1|X = x]$ in classification, smoother base procedures often perform better if the underlying regression or probability curve is a smooth function of continuous predictor variables (BY03).

Even when using trees, the question remains about the size of the tree. A guiding principle is as follows: take the smallest trees, i.e. trees with the smallest number k of terminal nodes, such that the class of linear combinations of k -node trees is sufficiently rich for the phenomenon to be modeled; of course, there is also here a trade-off between sample size and the complexity of the function class.

For example, when taking stumps with $k = 2$, the set of linear combinations of stumps is dense in (or “yields” the) set of additive functions (Bre04). In (FHT00), this is demonstrated from a more practical point of view. When taking trees with three terminal nodes ($k = 3$), the set of linear combinations of 3-node trees yields all second-order interaction functions. Thus, when aiming for consistent estimation of the full regression (or conditional class-probability) function, we should choose trees with $k = d + 1$ terminal nodes (in practice only if the sample size is “sufficiently large” in relation to d), (cf. (Bre04)).

Consistency of the AdaBoost algorithm is proved in (Jiao4), for example when using trees having $d + 1$ terminal nodes. More refined results are given in (MMZ02, ZY03) for modified boosting procedures with more general loss functions.

Interpretation

The main disadvantage from a statistical perspective is the lack of interpretation when boosting trees. This is in sharp contrast to boosting for linear, additive or interaction modeling. An approach to enhance interpretation is described in (Fri01).

16.3.9 Boosting and ℓ_1 -penalized Methods (Lasso)

Another method which does variable selection and variable amount of shrinkage is basis pursuit (CDS99) or Lasso (Tib96) which employs an ℓ_1 -penalty for the coefficients in the log-likelihood.

Interestingly, in case of linear least squares regression with a “positive cone condition” on the design matrix, an approximate equivalence of (a version of) L_2 Boosting and Lasso has been demonstrated in (EHJT03). More precisely, the set of boosting solutions, when using an (infinitesimally) small step size (see Sect. 16.3.3), over all the different boosting iterations, equals approximately the set of Lasso solutions when varying the ℓ_1 -penalty parameter. Moreover, the approximate set of boosting solutions can be computed very efficiently by the so-called least angle regression algorithm (EHJT03).

It is not clear to what extent this approximate equivalence between boosting and Lasso carries over to more general design matrices in linear regression or to other problems than regression with other loss functions. But the approximate equivalence in the above mentioned special case may help to understand boosting from a different perspective.

In the machine learning community, there has been a substantial focus on consistent estimation in the convex hull of function classes (cf. (Bar03, BJMo3, LV04)). For example, one may want to estimate a regression or probability function which can be written as

$$\sum_{k=1}^{\infty} w_k g_k(\cdot), \quad w_k \geq 0, \quad \sum_{k=1}^{\infty} w_k = 1,$$

where the $g_k(\cdot)$'s belong to a function class such as stumps or trees with a fixed number of terminal nodes. The quantity above is a convex combination of individual functions, in contrast to boosting which pursues linear combination of individual functions. By scaling, which is necessary in practice and theory (cf. (LV04)), one can actually look at this as a linear combination of functions whose coefficients satisfy $\sum_k w_k = \lambda$. This then represents an ℓ_1 -constraint as in Lasso, a relation which we have already outlined above.

16.3.10 Other References

Boosting, or functional gradient descent, has also been proposed for other settings than regression or classification, including survival analysis (Ben02), ordinal response problems (TH03) and high-multivariate financial time series (ABuo3, ABao3).

Support vector machines (cf. (Vap98, HTFo1, SSo2)) have become very popular in classification due to their good performance in a variety of data sets, similarly as boosting methods for classification. A connection between boosting and support vector machines has been made in (RZH03), suggesting also a modification of support vector machines to more sparse solutions (ZRHT03).

Acknowledgements. I would like to thank Marcel Dettling for some constructive comments.

References

- Allwein, E., Schapire, R., Singer, Y.: Reducing multiclass to binary: a unifying approach for margin classifiers. *J. Machine Learning Research* **1**, 113–141 (2001).
- Amit, Y., Geman, D.: Shape quantization and recognition with randomized trees. *Neural Computation* **9**, 1545–1588 (1997).
- Audrino F., Barone-Adesi G.: A multivariate FGD technique to improve VaR computation in equity markets. To appear in *Computational Management Science*.
- Audrino, F., Bühlmann, P.: Volatility estimation with functional gradient descent for very high-dimensional financial time series. *J. Computational Finance* **6**(3), 65–89 (2003).
- Bartlett, P.L.: Prediction algorithms: complexity, concentration and convexity. In: *Proceedings of the 13th IFAC Symposium on System Identification*, pp. 1507–1517 (2003).
- Bartlett, P.L., Jordan, M.I., McAuliffe, J.D.: Convexity, classification, and risk bounds. Technical Report 638, Dept. of Statistics, Univ. of Calif. (2003). Available from <http://www.stat.berkeley.edu/tech-reports/index.html>
- Bauer, E., Kohavi, R.: An empirical comparison of voting classification algorithms: bagging, boosting and variants. *Machine Learning*, **36**, 1545–1588 (1999).
- Benner, A.: Application of “aggregated classifiers” in survival time studies. In: *COMPSTAT 2002 - Proceedings in Computational Statistics - 15th Symposium held in Berlin* (Eds. Härdle, W. and Rönz, B.), Physika Verlag, Heidelberg (2002).
- Breiman, L.: Bagging predictors. *Machine Learning*, **24**, 123–140 (1996)
- Breiman, L.: Out-of-bag estimation. Technical Report (1996). Available from <ftp://ftp.stat.berkeley.edu/pub/users/breiman/>
- Breiman, L.: Arcing classifiers. *Annals of Statistics* **26**, 801–824 (1998).
- Breiman, L.: Prediction games & arcing algorithms. *Neural Computation* **11**, 1493–1517 (1999).
- Breiman, L.: Random Forests. Preprint. Available from <http://stat-www.berkeley.edu/users/breiman/rf.html>
- Breiman, L.: Population theory for boosting ensembles. To appear in *Annals of Statistics*, 32(1) (2004).
- Bühlmann, P.: Bagging, subbagging and bragging for improving some prediction algorithms. In: *Recent Advances and Trends in Nonparametric Statistics* (Eds. Akritas, M.G., Politis, D.N.), Elsevier (2003).
- Bühlmann, P.: Boosting for high-dimensional linear models. Preprint (2004). Available from <http://www.stat.math.ethz.ch/~buhlmann/bibliog.html>
- Bühlmann, P., Yu, B.: Discussion on Additive logistic regression: a statistical view of boosting (Auh. Friedman, J., Hastie, T., Tibshirani, R.) *Annals of Statistics* **28**, 377–386 (2000).

- Bühlmann, P., Yu, B.: Analyzing bagging. *Annals of Statistics* **30**, 927–961 (2002).
- Bühlmann, P., Yu, B.: Boosting with the L_2 loss: regression and classification. *J. American Statistical Association* **98**, 324–339 (2003).
- Buja, A., Stuetzle, W.: Observations on bagging. Preprint (2002). Available from <http://ljsavage.wharton.upenn.edu/~buja/>
- Bylander, T.: Estimating generalization error on two-class datasets using out-of-bag estimates. *Machine Learning* **48**, 287–297 (2002).
- Chen, S.S., Donoho, D.L., Saunders, M.A.: Atomic decomposition by basis pursuit. *SIAM J. Scientific Computing* **20**(1), 33–61 (1999).
- Chen, S.X., Hall, P.: Effects of bagging and bias correction on estimators defined by estimating equations. *Statistica Sinica* **13**, 97–109 (2003).
- Detting, M.: Bag-Boosting for tumor classification. In preparation (2004).
- Detting, M., Bühlmann, P.: Boosting for tumor classification with gene expression data. *Bioinformatics* **19**(9), 1061–1069 (2003).
- Borra, S., Di Ciaccio, A.: Improving nonparametric regression methods by bagging and boosting. *Computational Statistics & Data Analysis* **38**, 407–420 (2002).
- Dudoit, S., Fridlyand, J.: Bagging to improve the accuracy of a clustering procedure. *Bioinformatics* **19**(9), 1090–1099 (2003).
- Efron, B., Tibshirani, R.: The problem of regions. *Annals of Statistics* **26**, 1687–1718 (1998).
- Efron, B., Hastie, T., Johnstone, I., Tibshirani, R.: Least angle regression. To appear in *Annals of Statistics*, 32(2) (2004).
- Freund, Y. (1995): Boosting a weak learning algorithm by majority. *Information and Computation* **121**, 256–285 (1995).
- Freund, Y., Schapire, R.E.: Experiments with a new boosting algorithm. In *Machine Learning: Proc. Thirteenth International Conference*, pp. 148–156. Morgan Kaufman, San Francisco (1996).
- Friedman, J.H.: Multivariate adaptive regression splines. *Annals of Statistics* **19**, 1–141 (with discussion) (1991).
- Friedman, J.H.: Greedy function approximation: a gradient boosting machine. *Annals of Statistics* **29**, 1189–1232 (2001).
- Friedman, J.H., Hastie, T., Tibshirani, R.: Additive logistic regression: a statistical view of boosting. *Annals of Statistics* **28**, 337–407 (with discussion) (2000).
- Hastie, T.J., Tibshirani, R.J.: *Generalized Additive Models*. Chapman & Hall, London (1990).
- Hastie, T., Tibshirani, R., Friedman, J.: *The Elements of Statistical Learning. Data Mining, Inference and Prediction*. Springer, New York (2001).
- Hothorn, T., Lausen, B.: Bundling classifiers by bagging trees. Preprint (2002). Available from <http://www.mathpreprints.com/math/Preprint/blausen/20021016/1/>
- Hurvich, C.M., Simonoff, J.S., Tsai, C.-L.: Smoothing parameter selection in non-parametric regression using an improved Akaike information criterion. *J. Royal Statistical Society, Series B*, **60**, 271–293 (1998).
- Jiang, W.: process consistency for AdaBoost. To appear in *Annals of Statistics*, 32(1) (2004).

- Lugosi, G., Vayatis, N. On the Bayes-risk consistency of regularized boosting methods. To appear in *Annals of Statistics*, 32(1) (2004).
- Mallat, S., Zhang, Z. (1993). Matching pursuits with time-frequency dictionaries. *IEEE Transactions Signal Processing* 41, 3397–3415 (1993).
- Mannor, S., Meir, R., Zhang, T.: The consistency of greedy algorithms for classification. *Proceedings COLT02*, Vol. 2375 of *LNAI*, pp. 319–333, Sydney, Springer (2002).
- Mason, L., Baxter, J., Bartlett, P., Frean, M.: Functional gradient techniques for combining hypotheses. In: *advances in Large Margin Classifiers* (Eds. Smola, A.J., Bartlett, P.J., Schölkopf, B., Schuurmans, D.). MIT Press, Cambridge, MA (2000).
- Politis, D.N., Romano, J.P., Wolf, M.: *Subsampling*. Springer, New York (1999).
- Ridgeway, G.: Looking for lumps: boosting and bagging for density estimation. *Computational Statistics and Data Analysis* 38(4), 379–392 (2002).
- Rosset, S., Zhu, J., Hastie, T. Margin maximizing loss functions. Accepted poster for NIPS (2003). Available from <http://www-stat.stanford.edu/~hastie/pub.htm>
- Schapire, R.E.: The strength of weak learnability. *Machine Learning* 5, 197–227 (1990).
- Schapire, R.E.: The boosting approach to machine learning: an overview. In: *MSRI Workshop on Nonlinear Estimation and Classification* (Eds. Denison, D.D., Hansen, M.H., Holmes, C.C., Mallick, B., Yu, B). Springer, New York (2002).
- Schölkopf, B., Smola, A.J.: *Learning with Kernels*. MIT Press, Cambridge (2002).
- Tibshirani, R.: Regression shrinkage and selection via the lasso. *J. Royal Statistical Society, Series B*, 58, 267–288 (1996).
- Tukey, J.W.: *Exploratory data analysis*. Addison-Wesley, Reading, MA (1977).
- Tutz, G., Hechenbichler, K.: *Aggregating Classifiers With Ordinal Response Structure*, SFB 386 Discussion Paper No. 359 (2003). Available from <http://www.stat.uni-muenchen.de/sfb386/>
- Vapnik, V.N.: *Statistical Learning Theory*. Wiley, New York (1998).
- Wahba, G.: *Spline Models for Observational Data*. Society for Industrial and Applied Mathematics (1990).
- Zhang, T., Yu, B.: Boosting with early stopping: convergence and consistency. Technical Report 635, Dept. of Statistics, Univ. of Calif., Berkeley (2003). Available from <http://www.stat.berkeley.edu/users/binyu/publications.html>
- Zhu, J., Rosset, S., Hastie, T., Tibshirani, R.: 1-norm support vector machines. Accepted spotlight poster for NIPS (2003). Available from <http://www-stat.stanford.edu/~hastie/pub.htm>

Part IV
Selected Applications



Computationally Intensive Value at Risk Calculations

IV.1

Rafał Weron

1.1	<i>Introduction</i>	912
1.2	<i>Stable Distributions</i>	915
	Characteristic Function Representation	917
	Computation of Stable Density and Distribution Functions	917
	Simulation of α -stable Variables	921
	Estimation of Parameters	922
	Are Asset Returns α -stable?	928
	Truncated Stable Distributions	930
1.3	<i>Hyperbolic Distributions</i>	932
	Simulation of Generalized Hyperbolic Variables	936
	Estimation of Parameters	938
	Are Asset Returns NIG Distributed?	940
1.4	<i>Value at Risk, Portfolios and Heavy Tails</i>	942

Introduction

Market risks are the prospect of financial losses – or gains – due to unexpected changes in market prices and rates. Evaluating the exposure to such risks is nowadays of primary concern to risk managers in financial and non-financial institutions alike. Until late 1980s market risks were estimated through gap and duration analysis (interest rates), portfolio theory (securities), sensitivity analysis (derivatives) or “what-if” scenarios. However, all these methods either could be applied only to very specific assets or relied on subjective reasoning.

Since the early 1990s a commonly used market risk estimation methodology has been the Value at Risk (VaR). A VaR measure is the highest possible loss L incurred from holding the current portfolio over a certain period of time at a given confidence level (Dowd, 2002; Franke, Härdle and Stahl, 2000; Jorion, 2000):

$$\mathbb{P}(L > \text{VaR}) \leq 1 - c ,$$

where c is the confidence level, typically 95%, 97.5% or 99%. By convention, $L = -\Delta X(\tau)$, where $\Delta X(\tau)$ is the relative change (return) in portfolio value over the time horizon τ . Hence, large values of L correspond to large losses (or large negative returns).

The VaR figure has two important characteristics: (1) it provides a common consistent measure of risk across different positions and risk factors and (2) it takes into account the correlations or dependencies between different risk factors. Because of its intuitive appeal and simplicity, it is no surprise that in a few years Value at Risk has become the standard risk measure used around the world. However, VaR has a number deficiencies, among them the non-subadditivity – a sum of VaR’s of two portfolios can be smaller than the VaR of the combined portfolio. To cope with these shortcomings, Artzner et al. (1999) proposed an alternative measure that satisfies the assumptions of a coherent, i.e. an adequate, risk measure. The Expected Shortfall (ES), also called Expected Tail Loss or Conditional VaR, is the expected value of the losses in excess of VaR:

$$\text{ES} = \mathbb{E}(L|L > \text{VaR}) .$$

It is interesting to note, that – although new to the finance industry – Expected Shortfall has been familiar to insurance practitioners for a long time. It is very similar to the mean excess function which is used to characterize claim size distributions (Burnecki et al., 2004).

The essence of the VaR and ES computations is estimation of low quantiles in the portfolio return distributions. Hence, the performance of market risk measurement methods depends on the quality of distributional assumptions on the underlying risk factors. Many of the concepts in theoretical and empirical finance developed over the past decades – including the classical portfolio theory, the Black–Scholes–Merton option pricing model and even the RiskMetrics variance-covariance approach to VaR – rest upon the assumption that asset returns follow a normal distribution. But is this assumption justified by empirical data?

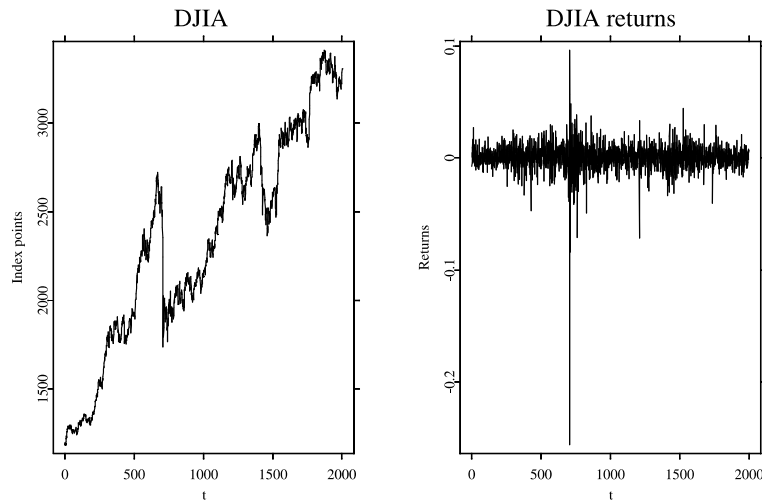


Figure 1.1. DJIA daily closing values X_t (left panel) and daily returns $\log(X_{t+1}/X_t)$ (right panel) from the period January 2, 1985 – November 30, 1992. Note, that this period includes Black Monday, the worst stock market crash in Wall Street history. On October 19, 1987 DJIA lost 508 points or 22.6% of its value (Q: CSAfin01)

No, it is not! It has been long known that asset returns are not normally distributed. Rather, the empirical observations exhibit excess kurtosis (fat tails). The Dow Jones Industrial Average (DJIA) index is a prominent example, see Fig. 1.1 where the index itself and its returns (or log-returns) are depicted. In Fig. 1.2 we plotted the empirical distribution of the DJIA index. The contrast with the Gaussian law is striking. This heavy tailed or leptokurtic character of the distribution of price changes has been repeatedly observed in various markets and may be quantitatively measured by the kurtosis in excess of 3, a value obtained for the normal distribution (Bouchaud and Potters, 2000; Carr et al., 2002; Guillaume et al., 1997; Mantegna and Stanley, 1995; Rachev and Mittnik, 2000). In Fig. 1.2 we also plotted vertical lines representing the Gaussian and empirical daily VaR estimates at the $c = 95\%$ and 99% confidence levels. They depict a typical situation encountered in financial markets. The Gaussian model overestimates the VaR number at the 95% confidence level and underestimates it at the 99% confidence level.

These VaR estimates are used here only for illustrative purposes and correspond to a one day VaR of a virtual portfolio consisting of one long position in the DJIA index. Note, that they are equal to the absolute value of the 5% and 1% quantiles, respectively. Hence, calculating the VaR number reduces to finding the $(1 - c)$ quantile. The empirical $(1 - c)$ quantile is obtained by taking the k th smallest value of the sample, where k is the smallest integer greater or equal to the length of the sample times $(1 - c)$. The Gaussian $(1 - c)$ quantile is equal to $F^{-1}(1 - c)$, where F is the normal distribution function. Since algorithms for evaluating the inverse of the Gaussian distribution function are implemented in practically any computing environment, calculating the quantile is straightforward.

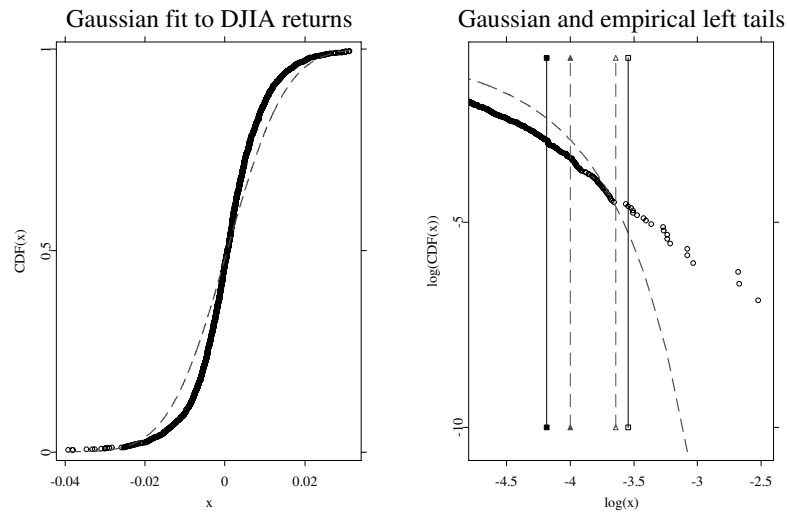


Figure 1.2. Gaussian (*dashed line*) fit to the DJIA daily returns (*circles*) empirical cumulative distribution function from the period January 2, 1985 – November 30, 1992. For better exposition of the fit in the central part of the distribution ten largest and ten smallest returns are not illustrated in the *left panel*. The *right panel* is a magnification of the left tail fit on a double logarithmic scale clearly showing the discrepancy between the data and the normal distribution. *Vertical lines* represent the Gaussian (*dashed line*) and empirical (*solid line*) VaR estimates at the 95% (*filled triangles and squares*) and 99% (*hollow triangles and squares*) confidence levels (Q: CSAfin02)

Interestingly, the problem of the underestimation of risk by the Gaussian distribution has been dealt with by the regulators in an ad hoc way. The Basle Committee on Banking Supervision (1995) suggested that for the purpose of determining minimum capital reserves financial institutions use a ten day VaR at the $c = 99\%$ confidence level multiplied by a safety factor $s \in [3, 4]$, with the exact value of s depending on the past performance of the model. It has been argued by Stahl (1997) and Danielsson, Hartmann and De Vries (1998) that the range of the safety factor comes from the heavy-tailed nature of the returns distribution. Indeed, if we assume that the asset returns distribution is symmetric and has finite variance σ^2 then from Chebyshev's inequality (Laha and Rohatgi, 1979) we obtain $\mathbb{P}(L \geq \varepsilon) \leq \sigma^2/2\varepsilon^2$, where L represents the random loss over the specified time horizon. So if we want to calculate the upper bound for a 99% VaR, setting $\sigma^2/2\varepsilon^2 = 1\%$ yields $\varepsilon = 7.07\sigma$, which in turn implies that $\text{VaR}_{99\%} \leq 7.07\sigma$. However, if we assumed a Gaussian distribution of returns then we would have $\text{VaR}_{99\%} \leq 2.33\sigma$, which is roughly three times lower than the bound obtained for a heavy-tailed, finite variance distribution.

Having said this much about the inadequacy of the Gaussian distribution for financial modeling and risk management we have no other choice but offer some heavy-tailed alternatives. We have to mention, though, that all distributional classes described in this chapter present computational challenge. Large parts of the text

are thus devoted to numerical issues. In Sect. 1.2 we deal with the historically earliest alternative – the stable laws and briefly characterize their recent generalizations – the so-called truncated stable distributions. In Sect. 1.3 we study the class of generalized hyperbolic laws. Finally, in Sect. 1.4 we introduce the notion of copulas and discuss the relation between VaR, asset portfolios and heavy tails.

All theoretical results are illustrated by empirical examples which utilize the quantlets (i.e. functions) of the XploRe computing environment (Härdle, Klinke and Müller, 2000). For reference, figure captions include names of the corresponding quantlets (Q). The reader of this chapter may therefore repeat and modify at will all the presented examples via the local XploRe Quantlet Server (XQS) without having to buy additional software. Such XQ Servers may be downloaded freely from the XploRe website <http://www.xplo-re-stat.de>. Currently, no other statistical computing environment offers a complete coverage of the issues discussed in this chapter. However, when available links to third-party libraries and specialized software are also provided.

Stable Distributions

1.2

It is often argued that financial asset returns are the cumulative outcome of a vast number of pieces of information and individual decisions arriving almost continuously in time (McCulloch, 1996; Rachev and Mittnik, 2000). As such, since the pioneering work of Louis Bachelier in 1900, they have been modeled by the Gaussian distribution. The strongest statistical argument for it is based on the Central Limit Theorem, which states that the sum of a large number of independent, identically distributed variables from a finite-variance distribution will tend to be normally distributed. However, financial asset returns usually have heavier tails.

In response to the empirical evidence Mandelbrot (1963) and Fama (1965) proposed the stable distribution as an alternative model. There are at least two good reasons for modeling financial variables using stable distributions. Firstly, they are supported by the generalized Central Limit Theorem, which states that stable laws are the only possible limit distributions for properly normalized and centered sums of independent, identically distributed random variables (Laha and Rohatgi, 1979). Secondly, stable distributions are leptokurtic. Since they can accommodate the fat tails and asymmetry, they fit empirical distributions much better.

Stable laws – also called α -stable, stable Paretian or Lévy stable – were introduced by Lévy (1925) during his investigations of the behavior of sums of independent random variables. A sum of two independent random variables having an α -stable distribution with index α is again α -stable with the same index α . This invariance property, however, does not hold for different α 's.

The α -stable distribution requires four parameters for complete description: an index of stability $\alpha \in (0, 2]$ also called the tail index, tail exponent or characteristic exponent, a skewness parameter $\beta \in [-1, 1]$, a scale parameter $\sigma > 0$

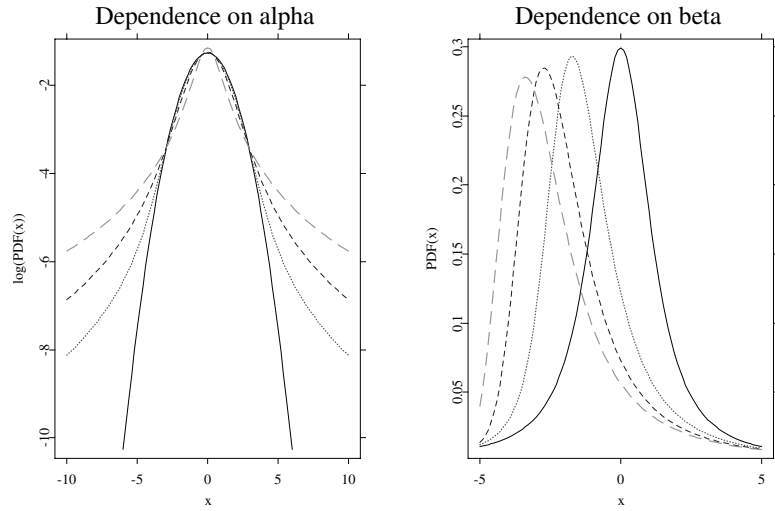


Figure 1.3. A semilog plot of symmetric ($\beta = \mu = 0$) α -stable probability density functions for $\alpha = 2$ (solid), 1.8 (dotted), 1.5 (dashed) and 1 (long-dashed) showing the dependence on the tail exponent (left panel). The Gaussian ($\alpha = 2$) density forms a parabola and is the only α -stable density with exponential tails. A plot of α -stable probability density functions for $\alpha = 1.2$ and $\beta = 0$ (solid), 0.5 (dotted), 0.8 (dashed) and 1 (long-dashed) showing the dependence on the skewness parameter (right panel) (Q: STFstab01, STFstab02)

and a location parameter $\mu \in \mathbb{R}$. The tail exponent α determines the rate at which the tails of the distribution taper off, see the left panel of Fig. 1.3. When $\alpha = 2$, a Gaussian distribution results. When $\alpha < 2$, the variance is infinite and the tails are asymptotically equivalent to a Pareto law, i.e. they exhibit a power-law behavior. More precisely, using a central limit theorem type argument it can be shown that (Janicki and Weron, 1994a; Samorodnitsky and Taqqu, 1994):

$$\begin{cases} \lim_{x \rightarrow \infty} x^\alpha \mathbb{P}(X > x) = C_\alpha(1 + \beta)\sigma^\alpha, \\ \lim_{x \rightarrow \infty} x^\alpha \mathbb{P}(X < -x) = C_\alpha(1 - \beta)\sigma^\alpha, \end{cases} \quad (1.1)$$

where:

$$C_\alpha = \left(2 \int_0^\infty x^{-\alpha} \sin(x) dx \right)^{-1} = \frac{1}{\pi} \Gamma(\alpha) \sin \frac{\pi\alpha}{2}.$$

When $\alpha > 1$, the mean of the distribution exists and is equal to μ . In general, the p th moment of a stable random variable is finite if and only if $p < \alpha$. When the skewness parameter β is positive, the distribution is skewed to the right, i.e. the right tail is thicker, see the right panel of Fig. 1.3. When it is negative, it is skewed to the left. When $\beta = 0$, the distribution is symmetric about μ . As α approaches 2, β loses its effect and the distribution approaches the Gaussian

distribution regardless of β . The last two parameters, σ and μ , are the usual scale and location parameters, i.e. σ determines the width and μ the shift of the mode (the peak) of the distribution.

Characteristic Function Representation

1.2.1

From a practitioner's point of view the crucial drawback of the stable distribution is that, with the exception of three special cases, its probability density function (PDF) and cumulative distribution function (CDF) do not have closed form expressions. These exceptions include the well known Gaussian ($\alpha = 2$) law, whose density function is given by:

$$f_G(x) = \frac{1}{\sqrt{2\pi}\sigma} \exp \left\{ -\frac{(x - \mu)^2}{2\sigma^2} \right\}, \quad (1.2)$$

and the lesser known Cauchy ($\alpha = 1, \beta = 0$) and Lévy ($\alpha = 0.5, \beta = 1$) laws.

Hence, the α -stable distribution can be most conveniently described by its characteristic function $\phi(t)$ – the inverse Fourier transform of the PDF. However, there are multiple parameterizations for α -stable laws and much confusion has been caused by these different representations. The variety of formulas is caused by a combination of historical evolution and the numerous problems that have been analyzed using specialized forms of the stable distributions. The most popular parameterization of the characteristic function of $X \sim S_\alpha(\sigma, \beta, \mu)$, i.e. an α -stable random variable with parameters α, σ, β and μ , is given by (Samorodnitsky and Taqqu, 1994; Weron, 1996):

$$\log \phi(t) = \begin{cases} -\sigma^\alpha |t|^\alpha \left\{ 1 - i\beta \operatorname{sign}(t) \tan \frac{\pi\alpha}{2} \right\} + i\mu t, & \alpha \neq 1, \\ -\sigma |t| \left\{ 1 + i\beta \operatorname{sign}(t) \frac{2}{\pi} \log |t| \right\} + i\mu t, & \alpha = 1. \end{cases} \quad (1.3)$$

Note, that the traditional scale parameter σ of the Gaussian distribution is not the same as σ in the above representation. A comparison of formulas (1.2) and (1.3) yields the relation: $\sigma_{\text{Gaussian}} = \sqrt{2}\sigma$.

For numerical purposes, it is often useful to use Nolan's (1997) parameterization:

$$\log \phi_0(t) = \begin{cases} -\sigma^\alpha |t|^\alpha \left\{ 1 + i\beta \operatorname{sign}(t) \tan \frac{\pi\alpha}{2} [(\sigma |t|)^{1-\alpha} - 1] \right\} + i\mu_0 t, & \alpha \neq 1, \\ -\sigma |t| \left\{ 1 + i\beta \operatorname{sign}(t) \frac{2}{\pi} \log(\sigma |t|) \right\} + i\mu_0 t, & \alpha = 1. \end{cases} \quad (1.4)$$

The $S_\alpha^0(\sigma, \beta, \mu_0)$ representation is a variant of Zolotarev's 1986 (M)-parameterization, with the characteristic function and hence the density and the distribution function jointly continuous in all four parameters, see Fig. 1.4. In particular, percentiles and convergence to the power-law tail vary in a continuous way as α

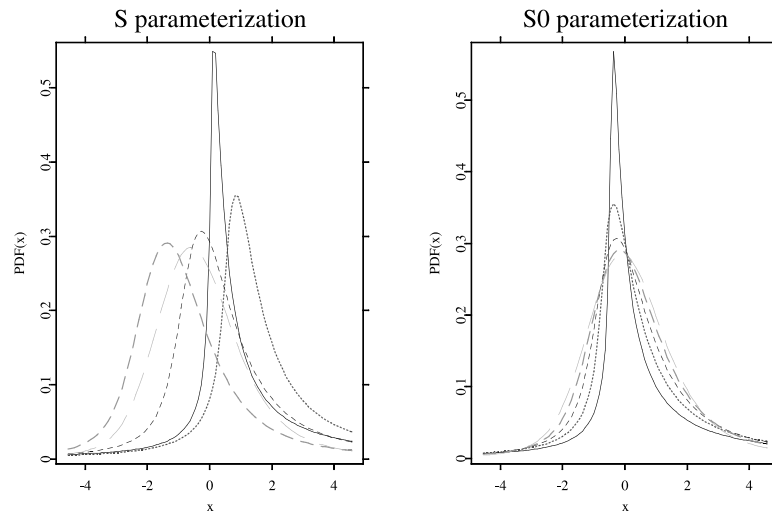


Figure 1.4. Comparison of S and S^0 parameterizations: α -stable probability density functions for $\beta = 0.5$ and $\alpha = 0.5$ (solid), 0.75 (dotted), 1 (short-dashed), 1.25 (dashed) and 1.5 (long-dashed) (Q: STFstab04)

and β vary. The location parameters of the two representations are related by $\mu = \mu_0 - \beta\sigma \tan(\pi\alpha/2)$ for $\alpha \neq 1$ and $\mu = \mu_0 - \beta\sigma(2/\pi) \log \sigma$ for $\alpha = 1$.

Computation of Stable Density and Distribution Functions

1.2.2

The lack of closed form formulas for most stable densities and distribution functions has negative consequences. Numerical approximation or direct numerical integration have to be used, leading to a drastic increase in computational time and loss of accuracy. Of all the attempts to be found in the literature a few are worth mentioning. DuMouchel (1971) developed a procedure for approximating the stable distribution function using Bergström's (1952) series expansion. Depending on the particular range of α and β , Holt and Crow (1973) combined four alternative approximations to compute the stable density function. Both algorithms are computationally intensive and time consuming, making maximum likelihood estimation a nontrivial task, even for modern computers. Recently, two other techniques have been proposed.

Mittnik, Doganoglu and Chenyao (1999) exploited the density function – characteristic function relationship and applied the fast Fourier transform (FFT). However, for data points falling between the equally spaced FFT grid nodes an interpolation technique has to be used. The authors suggested that linear interpolation suffices in most practical applications, see also Rachev and Mittnik (2000). Taking a larger number of grid points increases accuracy, however, at the expense of higher computational burden. Setting the number of grid points to $N = 2^{13}$

and the grid spacing to $h = 0.01$ allows to achieve comparable accuracy to the direct integration method (see below), at least for a range of α 's typically found for financial data ($1.6 < \alpha < 1.9$). As for the computational speed, the FFT based approach is faster for large samples, whereas the direct integration method favors small data sets since it can be computed at any arbitrarily chosen point. Mittnik, Doganoglu and Chenyao (1999) report that for $N = 2^{13}$ the FFT based method is faster for samples exceeding 100 observations and slower for smaller data sets.

We must stress, however, that the FFT based approach is not as universal as the direct integration method – it is efficient only for large alpha's and only as far as the probability density function calculations are concerned. When computing the cumulative distribution function the former method must numerically integrate the density, whereas the latter takes the same amount of time in both cases.

The direct integration method, proposed by Nolan (1997, 1999) consists of a numerical integration of Zolotarev's (1986) formulas for the density or the distribution function. To save space we state only the formulas for the probability density function. Complete formulas can be also found in Borak, Härdle and Weron (2004).

Set $\zeta = -\beta \tan \frac{\pi\alpha}{2}$. Then the density $f(x; \alpha, \beta)$ of a standard α -stable random variable in representation S^0 , i.e. $X \sim S^0_\alpha(1, \beta, 0)$, can be expressed as (note, that Zolotarev (1986, Sect. 2.2) used yet another parametrization):

– when $\alpha \neq 1$ and $x > \zeta$:

$$f(x; \alpha, \beta) = \frac{\alpha(x - \zeta)^{\frac{1}{\alpha}-1}}{\pi |\alpha - 1|} \int_{-\theta_0}^{\frac{\pi}{2}} V(\theta; \alpha, \beta) \exp \left\{ -(x - \zeta)^{\frac{\alpha}{\alpha-1}} V(\theta; \alpha, \beta) \right\} d\theta ,$$

– when $\alpha \neq 1$ and $x = \zeta$:

$$f(x; \alpha, \beta) = \frac{\Gamma\left(1 + \frac{1}{\alpha}\right) \cos(\xi)}{\pi(1 + \zeta^2)^{\frac{1}{2\alpha}}} ,$$

– when $\alpha \neq 1$ and $x < \zeta$:

$$f(x; \alpha, \beta) = f(-x; \alpha, -\beta) ,$$

– when $\alpha = 1$:

$$f(x; 1, \beta) = \begin{cases} \frac{1}{2|\beta|} e^{\frac{\pi x}{2\beta}} \int_{-\frac{\pi}{2}}^{\frac{\pi}{2}} V(\theta; 1, \beta) \exp \left\{ -e^{\frac{\pi x}{2\beta}} V(\theta; 1, \beta) \right\} d\theta , & \beta \neq 0 , \\ \frac{1}{\pi(1 + x^2)} , & \beta = 0 , \end{cases}$$

where

$$\xi = \begin{cases} \frac{1}{\alpha} \arctan(-\zeta) , & \alpha \neq 1 , \\ \frac{\pi}{2} , & \alpha = 1 , \end{cases} \tag{1.5}$$

and

$$V(\theta; \alpha, \beta) = \begin{cases} (\cos \alpha \xi)^{\frac{1}{\alpha-1}} \left(\frac{\cos \theta}{\sin \alpha(\xi + \theta)} \right)^{\frac{\alpha}{\alpha-1}} \frac{\cos \{\alpha \xi + (\alpha-1)\theta\}}{\cos \theta}, & \alpha \neq 1, \\ \frac{2}{\pi} \left(\frac{\frac{\pi}{2} + \beta \theta}{\cos \theta} \right) \exp \left\{ \frac{1}{\beta} \left(\frac{\pi}{2} + \beta \theta \right) \tan \theta \right\}, & \alpha = 1, \beta \neq 0. \end{cases}$$

XploRe offers the direct integration method through the `cdfstab` and `pdfstab` quantlets, see Cizek, Härdle and Weron (2004) for a thorough exposition of quantlets related to stable distributions. On a PC equipped with a Centrino 1.6 GHz processor the calculation of the stable distribution or density function at 1000 points takes about 1.2 seconds. As default, the integrals found in the above formulas are numerically integrated using 2000 subintervals. These computational times can be improved when using a numerically more efficient environment. For example, the program STABLE (downloadable from John Nolan's web page: <http://academic2.american.edu/~jpnolan/stable/stable.html>) needs about 0.9 seconds for performing corresponding calculations. It was written in Fortran and calls several external IMSL routines, see Nolan (1997) for details. Apart from speed, the STABLE program also exhibits higher relative accuracy (for default tolerance settings in both programs): about 10^{-13} compared to 10^{-10} for values used in typical financial applications (like approximating asset return distributions). Naturally, the accuracy of both programs can be increased at the cost of computational time.

It is interesting to note, that currently no other statistical computing environment offers the computation of stable density and distribution functions in its standard release. Users have to rely on third-party libraries or commercial products. A few are worth mentioning. John Nolan offers the STABLE program in library form through Robust Analysis Inc., see <http://www.robustanalysis.com>. This library (in C, Visual Basic or Fortran) provides interfaces to Matlab, S-plus (or its GNU version – R) and Mathematica. Diethelm Würtz has developed Rmetrics, an open source collection of software packages for S-plus/R, which may be useful for teaching computational finance, see <http://www.itp.phys.ethz.ch/econophysics/R/>. Stable PDF and CDF calculations are performed using the direct integration method, with the integrals being computed by R's function `integrate`. Interestingly, for symmetric stable distributions Rmetrics utilizes McCulloch's (1998) approximation, which was obtained by interpolating between the complements of the Cauchy and Gaussian CDFs in a transformed space. For $\alpha > 0.92$ the absolute precision of the stable PDF and CDF approximation is 10^{-4} . The FFT based approach is utilized in Cognity, a commercial risk management platform that offers derivatives pricing and portfolio optimization based on the assumption of stably distributed returns, see <http://www.finanalytica.com>.

Simulation of α -stable Variables

1.2.3

The complexity of the problem of simulating sequences of α -stable random variables stems from the fact that there are no analytic expressions for the inverse $F^{-1}(x)$ nor the cumulative distribution function $F(x)$. All standard approaches like the rejection or the inversion methods would require tedious computations. See Chap. II.2 for a review of non-uniform random number generation techniques.

A much more elegant and efficient solution was proposed by Chambers, Mallows and Stuck (1976). They noticed that a certain integral formula derived by Zolotarev (1964) yielded the following algorithm:

- generate a random variable U uniformly distributed on $(-\frac{\pi}{2}, \frac{\pi}{2})$ and an independent exponential random variable W with mean 1;
- for $\alpha \neq 1$ compute:

$$X = (1 + \zeta^2)^{\frac{1}{2\alpha}} \frac{\sin\{\alpha(U + \xi)\}}{\{\cos(U)\}^{1/\alpha}} \left[\frac{\cos\{U - \alpha(U + \xi)\}}{W} \right]^{\frac{1-\alpha}{\alpha}}, \quad (1.6)$$

- for $\alpha = 1$ compute:

$$X = \frac{1}{\xi} \left\{ \left(\frac{\pi}{2} + \beta U \right) \tan U - \beta \log \left(\frac{\frac{\pi}{2} W \cos U}{\frac{\pi}{2} + \beta U} \right) \right\}, \quad (1.7)$$

where ξ is given by (1.5). This algorithm yields a random variable $X \sim S_\alpha(1, \beta, 0)$, in representation (1.3). For a detailed proof see Weron (1996).

Given the formulas for simulation of a standard α -stable random variable, we can easily simulate a stable random variable for all admissible values of the parameters α, σ, β and μ using the following property: if $X \sim S_\alpha(1, \beta, 0)$ then

$$Y = \begin{cases} \sigma X + \mu, & \alpha \neq 1, \\ \sigma X + \frac{2}{\pi} \beta \sigma \log \sigma + \mu, & \alpha = 1, \end{cases}$$

is $S_\alpha(\sigma, \beta, \mu)$. It is interesting to note that for $\alpha = 2$ (and $\beta = 0$) the Chambers–Mallows–Stuck method reduces to the well known Box–Muller (1958) algorithm for generating Gaussian random variables (Janicki and Weron, 1994b).

Many other approaches have been proposed in the literature, including application of Bergström (1952) and LePage (LePage, Woodrooffe and Zinn, 1981) series expansions, see Mantegna (1994) and Janicki and Kokoszka (1992), respectively. However, this method is regarded as the fastest and the most accurate. In XploRe the algorithm is implemented in the `rndstab` quantlet. On a PC equipped with a Centrino 1.6 GHz processor one million variables are generated in about 3 seconds, compared to about 0.4 seconds for one million standard normal random variables obtained via the Box–Muller algorithm (`normal2`). Because of its unquestioned superiority and relative simplicity, the Chambers–Mallows–Stuck method is implemented in some statistical computing environments (e.g. the `rstable` function in S-plus/R) even if no other routines related to stable distributions are provided.

1.2.4 Estimation of Parameters

The estimation of stable law parameters is in general severely hampered by the lack of known closed-form density functions for all but a few members of the stable family. Numerical approximation or direct numerical integration are non-trivial and burdensome from a computational point of view. As a consequence, the maximum likelihood (ML) estimation algorithm based on such approximations is difficult to implement and time consuming for samples encountered in modern finance. However, there are also other numerical methods that have been found useful in practice and are discussed in this section.

All presented methods work quite well assuming that the sample under consideration is indeed α -stable. Since there are no formal tests for assessing the α -stability of a data set we suggest to first apply the “visual inspection” and tail exponent estimators to see whether the empirical densities resemble those of α -stable laws (Borak, Härdle and Weron, 2004; Weron, 2001).

Given a sample x_1, \dots, x_n of independent and identically distributed (i.i.d.) $S_\alpha(\sigma, \beta, \mu)$ observations, in what follows, we provide estimates $\hat{\alpha}$, $\hat{\sigma}$, $\hat{\beta}$ and $\hat{\mu}$ of all four stable law parameters. We start the discussion with the simplest, fastest and ... least accurate quantile methods, then develop the slower, yet much more accurate sample characteristic function methods and, finally, conclude with the slowest but most accurate maximum likelihood approach.

Sample Quantile Methods

Fama and Roll (1971) provided very simple estimates for parameters of symmetric ($\beta = 0, \mu = 0$) stable laws with $\alpha > 1$. They proposed to estimate σ by:

$$\hat{\sigma} = \frac{\hat{x}_{0.72} - \hat{x}_{0.28}}{1.654}, \quad (1.8)$$

where x_f denotes the f -th population quantile, so that $S_\alpha(\sigma, \beta, \mu)(x_f) = f$. As McCulloch (1986) noticed, Fama and Roll based their estimator of σ on the fortuitous observation that $(x_{0.72} - x_{0.28})/\sigma$ lies within 0.4% of 1.654 for all $1 < \alpha \leq 2$, when $\beta = 0$. This enabled them to estimate σ by (1.8) with less than 0.4% asymptotic bias without first knowing α . However, when $\beta \neq 0$, the search for an invariant range such as the one they found becomes futile.

The characteristic exponent α , on the other hand, can be estimated from the tail behavior of the distribution. Fama and Roll suggested to take $\hat{\alpha}$ satisfying:

$$S_{\hat{\alpha}}\left(\frac{\hat{x}_f - \hat{x}_{1-f}}{2\hat{\sigma}}\right) = f. \quad (1.9)$$

They found that $f = 0.95, 0.96, 0.97$ worked best for estimating α . This method unnecessarily compounds the small asymptotic bias in the estimator of σ into the estimator of α .

For $1 < \alpha \leq 2$, the stable distribution has finite mean. Hence, the sample mean is a consistent estimate of the location parameter μ . However, a more robust estimate

is the $p\%$ truncated sample mean – the arithmetic mean of the middle p percent of the ranked observations. The 50% truncated mean is often suggested in the literature when the range of α is unknown.

Fama and Roll's (1971) method is simple but suffers from a small asymptotic bias in $\hat{\alpha}$ and $\hat{\sigma}$ and restrictions on α and β . McCulloch (1986) generalized and improved the quantile method. He analyzed stable law quantiles and provided consistent estimators of all four stable parameters, with the restriction $\alpha \geq 0.6$, while retaining the computational simplicity of Fama and Roll's method. After McCulloch define:

$$v_\alpha = \frac{x_{0.95} - x_{0.05}}{x_{0.75} - x_{0.25}} \quad \text{and} \quad v_\beta = \frac{x_{0.95} + x_{0.05} - 2x_{0.50}}{x_{0.95} - x_{0.05}}. \quad (1.10)$$

Statistics v_α and v_β are functions of α and β only, i.e. they are independent of both σ and μ . This relationship may be inverted and the parameters α and β may be viewed as functions of v_α and v_β . Substituting v_α and v_β by their sample values and applying linear interpolation between values found in tables given in McCulloch (1986) yields estimators $\hat{\alpha}$ and $\hat{\beta}$.

Scale and location parameters, σ and μ , can be estimated in a similar way. However, due to the discontinuity of the characteristic function for $\alpha = 1$ and $\beta \neq 0$ in representation (1.3), this procedure is more complicated. We refer the interested reader to the original work of McCulloch (1986). This estimation technique is implemented in XploRe in the `stabcull` quantlet.

Sample Characteristic Function Methods

Given an i.i.d. random sample x_1, \dots, x_n of size n , define the sample characteristic function by:

$$\hat{\phi}(t) = \frac{1}{n} \sum_{j=1}^n \exp(itx_j).$$

Since $|\hat{\phi}(t)|$ is bounded by unity all moments of $\hat{\phi}(t)$ are finite and, for any fixed t , it is the sample average of i.i.d. random variables $\exp(itx_j)$. Hence, by the law of large numbers, $\hat{\phi}(t)$ is a consistent estimator of the characteristic function $\phi(t)$.

Press (1972) proposed a simple estimation method, called the method of moments, based on transformations of the characteristic function. From (1.3) we have for all α :

$$|\phi(t)| = \exp(-\sigma^\alpha |t|^\alpha). \quad (1.11)$$

Hence, $-\log |\phi(t)| = \sigma^\alpha |t|^\alpha$. Now, assuming $\alpha \neq 1$, choose two nonzero values of t , say $t_1 \neq t_2$. Then for $k = 1, 2$ we have:

$$-\log |\phi(t_k)| = \sigma^\alpha |t_k|^\alpha. \quad (1.12)$$

Solving these two equations for α and σ , and substituting $\hat{\phi}(t)$ for $\phi(t)$ yields:

$$\hat{\alpha} = \frac{\log \frac{\log |\hat{\phi}(t_1)|}{\log |\hat{\phi}(t_2)|}}{\log \left| \frac{t_1}{t_2} \right|}, \tag{1.13}$$

and

$$\log \hat{\sigma} = \frac{\log |t_1| \log(-\log |\hat{\phi}(t_2)|) - \log |t_2| \log(-\log |\hat{\phi}(t_1)|)}{\log \left| \frac{t_1}{t_2} \right|}. \tag{1.14}$$

In order to estimate β and μ we have to choose two nonzero values of t , say $t_3 \neq t_4$, and apply a similar trick to $\text{Im}\{\log \phi(t)\}$. The estimators are consistent since they are based upon estimators of $\phi(t)$, $\text{Im}\{\phi(t)\}$ and $\text{Re}\{\phi(t)\}$, which are known to be consistent. However, convergence to the population values depends on the choice of t_1, \dots, t_4 . The optimal selection of these values is problematic and still is an open question. The XploRe implementation of the method of moments (the `stabmom` quantlet) uses $t_1 = 0.2$, $t_2 = 0.8$, $t_3 = 0.1$, and $t_4 = 0.4$ as proposed by Koutrouvelis (1980) in his simulation study.

In the same paper Koutrouvelis presented a much more accurate regression-type method which starts with an initial estimate of the parameters and proceeds iteratively until some prespecified convergence criterion is satisfied. Each iteration consists of two weighted regression runs. The number of points to be used in these regressions depends on the sample size and starting values of α . Typically no more than two or three iterations are needed. The speed of the convergence, however, depends on the initial estimates and the convergence criterion.

The regression method is based on the following observations concerning the characteristic function $\phi(t)$. First, from (1.3) we can easily derive:

$$\log(-\log |\phi(t)|^2) = \log(2\sigma^\alpha) + \alpha \log |t|. \tag{1.15}$$

The real and imaginary parts of $\phi(t)$ are for $\alpha \neq 1$ given by:

$$\text{Re}\{\phi(t)\} = \exp(-|\sigma t|^\alpha) \cos \left[\mu t + |\sigma t|^\alpha \beta \text{sign}(t) \tan \frac{\pi\alpha}{2} \right],$$

and

$$\text{Im}\{\phi(t)\} = \exp(-|\sigma t|^\alpha) \sin \left[\mu t + |\sigma t|^\alpha \beta \text{sign}(t) \tan \frac{\pi\alpha}{2} \right].$$

The last two equations lead, apart from considerations of principal values, to:

$$\arctan \left(\frac{\text{Im}\{\phi(t)\}}{\text{Re}\{\phi(t)\}} \right) = \mu t + \beta \sigma^\alpha \tan \frac{\pi\alpha}{2} \text{sign}(t) |t|^\alpha. \tag{1.16}$$

Equation (1.15) depends only on α and σ and suggests that we estimate these parameters by regressing $y_k = \log(-\log|\hat{\phi}(t_k)|^2)$ on $w_k = \log|t_k|$ in the model:

$$y_k = m + \alpha w_k + \varepsilon_k, \tag{1.17}$$

where t_k is an appropriate set of real numbers, $m = \log(2\sigma^\alpha)$, and ε_k denotes an error term. Koutrouvelis (1980) proposed to use $t_k = \frac{\pi k}{25}, k = 1, 2, \dots, K$; with K ranging between 9 and 134 for different values of α and sample sizes.

Once $\hat{\alpha}$ and $\hat{\sigma}$ have been obtained and α and σ have been fixed at these values, estimates of β and μ can be obtained using (1.16). Next, the regressions are repeated with $\hat{\alpha}, \hat{\sigma}, \hat{\beta}$ and $\hat{\mu}$ as the initial parameters. The iterations continue until a prespecified convergence criterion is satisfied. Koutrouvelis proposed to use the Fama–Roll estimator (1.8) and the 25% truncated mean for initial estimates of σ and μ , respectively.

Kogon and Williams (1998) eliminated this iteration procedure and simplified the regression method. For initial estimation they applied McCulloch’s method, worked with the continuous representation (1.4) of the characteristic function instead of the classical one (1.3) and used a fixed set of only 10 equally spaced frequency points t_k . In terms of computational speed their method compares favorably to the original method of Koutrouvelis, see Table 1.1. It has a significantly better performance near $\alpha = 1$ and $\beta \neq 0$ due to the elimination of discontinuity of the characteristic function. However, it returns slightly worse results for other values of α . In XploRe both regression algorithms are implemented in the `stabreg` quantlet. An optional parameter lets the user choose between the original Koutrouvelis code and the Kogon–Williams modification.

Table 1.1. Comparison of McCulloch’s quantile technique, the method of moments, the regression approach of Koutrouvelis and the method of Kogon and Williams for 100 simulated samples of two thousand $S_{1,7}(0.005, 0.1, 0.001)$ random numbers each. Parameter estimates are mean values over 100 samples. Values of the Mean Absolute Percentage Error ($MAPE_\theta = \frac{1}{n} \sum_{i=1}^n |\hat{\theta} - \theta|/|\theta|$) are given in parentheses. In the last column average computational times for one sample of 2000 random variables are provided (on a PC equipped with a Centrino 1.6 GHz processor and running XploRe 4.6) (Q: CSAfin03)

Method	$\hat{\alpha}$	$\hat{\sigma}$	$\hat{\beta}$	$\hat{\mu}$	CPU time
McCulloch	1.7005 (2.60%)	0.0050 (2.16%)	0.1045 (110.72%)	0.0010 (22.01%)	0.025 s
Moments	1.9895 (17.03%)	0.0104 (107.64%)	0.0712 (969.57%)	0.0010 (33.56%)	0.015 s
Koutrouvelis	1.6988 (1.66%)	0.0050 (1.69%)	0.0989 (108.21%)	0.0010 (21.01%)	0.300 s
Kogon–Williams	1.6994 (1.95%)	0.0050 (1.77%)	0.0957 (110.59%)	0.0010 (21.14%)	0.085 s

A typical performance of the described estimators is summarized in Table 1.1, see also Fig. 1.5. McCulloch's quantile technique, the method of moments, the regression approach of Koutrouvelis and the method of Kogon and Williams were applied to 100 simulated samples of two thousand $S_{1.7}(0.005, 0.1, 0.001)$ random numbers each. The method of moments yielded the worst estimates, clearly outside any admissible error range. McCulloch's method came in next with acceptable results and computational time significantly lower than the regression approaches. On the other hand, both the Koutrouvelis and the Kogon–Williams implementations yielded good estimators with the latter performing considerably faster, but slightly less accurate. We have to say, though, that all methods had problems with estimating β . Like it or not, our search for the optimal estimation technique is not over yet. We have no other choice but turn to the last resort – the maximum likelihood method.

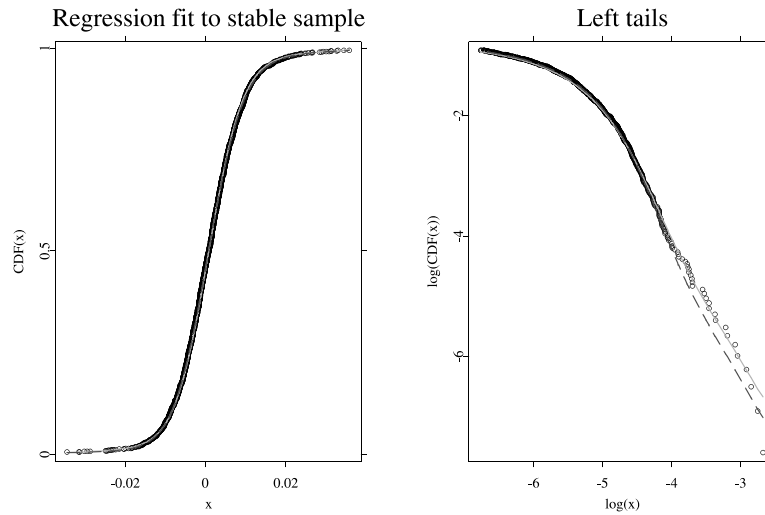


Figure 1.5. Regression fit (*dashed*), using Koutrouvelis' regression method, to 2000 simulated $S_{1.7}(0.005, 0.1, 0.001)$ random variables (*circles*). For comparison, the CDF of the original distribution is also plotted (*solid*). The *right panel* is a magnification of the left tail fit on a double logarithmic scale (Q: CSAf in04)

Maximum Likelihood Method

The maximum likelihood (ML) estimation scheme for α -stable distributions does not differ from that for other laws, at least as far as the theory is concerned. For a vector of observations $x = (x_1, \dots, x_n)$, the ML estimate of the parameter vector $\theta = (\alpha, \sigma, \beta, \mu)$ is obtained by maximizing the log-likelihood function:

$$L_{\theta}(x) = \sum_{i=1}^n \log \tilde{f}(x_i; \theta), \quad (1.18)$$

where $\tilde{f}(\cdot; \theta)$ is the stable density function. The tilde denotes the fact that, in general, we do not know the explicit form of the stable PDF and have to approximate it numerically. The ML methods proposed in the literature differ in the choice of the approximating algorithm. However, all of them have an appealing common feature – under certain regularity conditions the maximum likelihood estimator is asymptotically normal with the variance specified by the Fischer information matrix (DuMouchel, 1973). The latter can be approximated either by using the Hessian matrix arising in maximization or, as in Nolan (2001), by numerical integration.

Because of computational complexity there are only a few documented attempts of estimating stable law parameters via maximum likelihood. DuMouchel (1971) developed an approximate ML method, which was based on grouping the data set into bins and using a combination of means to compute the density (FFT for the central values of x and series expansions for the tails) to compute an approximate log-likelihood function. This function was then numerically maximized.

Applying Zolotarev's (1964) integral formulas, Brorsen and Yang (1990) formulated another approximate ML method, however, only for symmetric stable random variables. To avoid the discontinuity and nondifferentiability of the symmetric stable density function at $\alpha = 1$, the tail index α was restricted to be greater than one.

Much better, in terms of accuracy and computational time, are more recent maximum likelihood estimation techniques. Mittnik et al. (1999) utilized the FFT approach for approximating the stable density function, whereas Nolan (2001) used the direct integration method. Both approaches are comparable in terms of efficiency. The differences in performance are the result of different approximation algorithms, see Sect. 1.2.2.

As Ojeda (2001) observes, the ML estimates are almost always the most accurate, closely followed by the regression-type estimates, McCulloch's quantile method, and finally the method of moments. However, as we have already said in the introduction to this section, maximum likelihood estimation techniques are certainly the slowest of all the discussed methods. For example, ML estimation for a sample of 2000 observations using a gradient search routine which utilizes the direct integration method needs 221 seconds or about 3.7 minutes! The calculations were performed on a PC equipped with a Centrino 1.6 GHz processor and running STABLE ver. 3.13 (see also Sect. 1.2.2 where the program was briefly described). For comparison, the STABLE implementation of the Kogon-Williams algorithm performs the same calculations in only 0.02 seconds (the XploRe quantile `stabreg` needs roughly four times more time, see Table 1.1). Clearly, the higher accuracy does not justify the application of ML estimation in many real life problems, especially when calculations are to be performed on-line. For this reason the program STABLE also offers an alternative – a fast quasi ML technique. It quickly approximates stable densities using a 3-dimensional spline interpolation based on pre-computed values of the standardized stable density on a grid of (x, α, β) values. At the cost of a large array of coefficients, the interpolation is highly accurate over

Table 1.2. α -stable and Gaussian fits to 2000 returns of the Dow Jones Industrial Average (DJIA) index from the period January 2, 1985 – November 30, 1992. Values of the Anderson-Darling and Kolmogorov goodness-of-fit statistics suggest a much better fit of the 1.66-stable law. Empirical and model based (α -stable and Gaussian) VaR numbers at the 95% and 99% confidence levels are also given. The values in parentheses are the relative differences between model and empirical VaR estimates (Q: CSAfin05)

Parameters	α	σ	β	μ
α -stable fit	1.6596	0.0053	0.0823	0.0009
Gaussian fit		0.0115		0.0006
Test values	Anderson–Darling		Kolmogorov	
α -stable fit	1.0044		0.8641	
Gaussian fit	+ INF		4.5121	
VaR estimates ($\times 10^{-2}$)	95%		99%	
Empirical	1.5242		2.8922	
α -stable fit	1.3296	(12.77%)	2.7480	(4.98%)
Gaussian fit	1.8350	(20.39%)	2.6191	(9.44%)

most values of the parameter space and relatively fast – 0.26 seconds for a sample of 2000 observations.

1.2.5 Are Asset Returns α -stable?

In this paragraph we want to apply the discussed techniques to financial data. Due to limited space we have chosen only one estimation method – the regression approach of Koutrouvelis (1980), as it offers high accuracy at moderate computational time. We start the empirical analysis with the most prominent example – the Dow Jones Industrial Average (DJIA) index. The data set covers the period January 2, 1985 – November 30, 1992 and comprises 2000 returns. Recall, that this period includes the largest crash in Wall Street history – the Black Monday of October 19, 1987. Clearly the 1.66-stable law offers a much better fit to the DJIA returns than the Gaussian distribution, see Table 1.2. Its superiority, especially in the tails of the distribution, is even better visible in Fig. 1.6. In this figure we also plotted vertical lines representing the 1.66-stable, Gaussian and empirical daily VaR estimates at the $c = 95\%$ and 99% confidence levels. These estimates correspond to a one day VaR of a virtual portfolio consisting of one long position in the DJIA index. The stable VaR estimates are almost twice closer to the empirical estimates than the Gaussian ones, see Table 1.2.

Recall that calculating the VaR number reduces to finding the $(1 - c)$ quantile of a given distribution or equivalently to evaluating the inverse F^{-1} of the distribution function at $(1 - c)$. Unfortunately no simple algorithms for inverting the stable CDF are known. The `qfstab` quantlet of XploRe utilizes a simple binary search

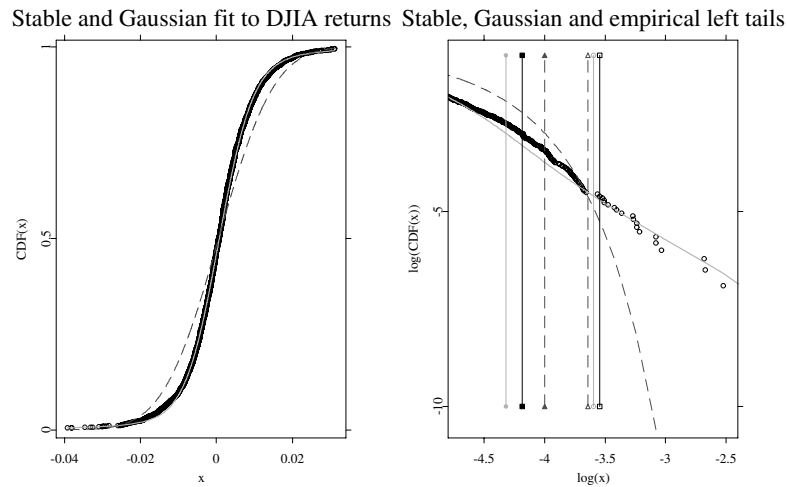


Figure 1.6. 1.66-stable (solid grey line) and Gaussian (dashed line) fits to the DJIA returns (circles) empirical cumulative distribution function from the period January 2, 1985 – November 30, 1992. For better exposition of the fit in the central part of the distribution ten largest and ten smallest returns are not illustrated in the left panel. The right panel is a magnification of the left tail fit on a double logarithmic scale. Vertical lines represent the 1.66-stable (solid grey line), Gaussian (dashed line) and empirical (solid line) VaR estimates at the 95% (filled circles, triangles and squares) and 99% (hollow circles, triangles and squares) confidence levels (Q: CSAfin05)

routine for large α 's and values near the mode of the distribution. In the extreme tails the approximate range of the quantile values is first estimated via the power law formula (1.1), then a binary search is conducted.

To make our statistical analysis more sound, we also compare both fits through Anderson–Darling and Kolmogorov test statistics (D’Agostino and Stephens, 1986). The former may be treated as a weighted Kolmogorov statistics which puts more weight to the differences in the tails of the distributions. Although no asymptotic results are known for the stable laws, approximate critical values for these goodness-of-fit tests can be obtained via the bootstrap technique (Borak, Härdle and Weron, 2004; Stute, Manteiga and Quindimil, 1993). In this chapter, though, we will not perform hypothesis testing and just compare the test values. Naturally, the lower the values the better the fit. The stable law seems to be tailor-cut for the DJIA index returns. But does it fit other asset returns as well?

The second analyzed data set comprises 2000 returns of the Deutsche Aktienindex (DAX) index from the period January 2, 1995 – December 5, 2002. Also in this case the α -stable law offers a much better fit than the Gaussian, see Table 1.3. However, the test statistics suggest that the fit is not as good as for the DJIA returns (observe that both data sets are of the same size and the test values in both cases can be compared). This can be also seen in Fig. 1.7. The left tail seems to drop off at some point and the very tail is largely overestimated by the stable distribution. At the same time it is better approximated by the Gaussian law. This results in

a surprisingly good fit of the daily 95% VaR by the Gaussian distribution, see Table 1.3, and an overestimation of the daily VaR estimate at the $c = 99\%$ confidence level by the 1.7-stable distribution. In fact, the latter is a rather typical situation. For a risk manager who likes to play safe this may not be a bad idea, as the stable law overestimates the risks and thus provides an upper limit of losses.

Table 1.3. α -stable and Gaussian fits to 2000 returns of the Deutsche Aktienindex (DAX) index from the period January 2, 1995 – December 5, 2002. Empirical and model based (α -stable and Gaussian) VaR numbers at the 95% and 99% confidence levels are also given (Q: CSAFin06)

Parameters	α	σ	β	μ
α -stable fit	1.7003	0.0088	-0.3179	-0.0002
Gaussian fit		0.0157		0.0004
Test values	Anderson–Darling		Kolmogorov	
α -stable fit	1.9149		1.1798	
Gaussian fit	16.4119		2.8197	
VaR estimates ($\times 10^{-2}$)	95%		99%	
Empirical	2.5731		4.5963	
α -stable fit	2.4296	(5.58%)	5.0982	(10.92%)
Gaussian fit	2.5533	(0.77%)	3.6260	(21.11%)

This example clearly shows that the α -stable distribution is not a panacea. Although it gives a very good fit to a number of empirical data sets, there surely are distributions that recover the characteristics of other data sets better. We devote the rest of this chapter to such alternative heavy tailed distributions. We start with a modification of the stable law and in Sect. 1.3 concentrate on the class of generalized hyperbolic distributions.

1.2.6 Truncated Stable Distributions

Mandelbrot's (1963) pioneering work on applying α -stable distributions to asset returns gained support in the first few years after its publication (Fama, 1965; Officer, 1972; Teichmoeller, 1971). Subsequent works, however, have questioned the stable distribution hypothesis (Akgiray and Booth, 1988; Blattberg and Gonedes, 1974). By the definition of the stability property, the sum of i.i.d. stable random variables is also stable. Thus, if short term asset returns are distributed according to a stable law, longer term returns should retain the same functional form. However, from the empirical data it is evident that as the time interval between price observations grows longer, the distribution of returns deviates from the short term heavy tailed distribution, and converges to the Gaussian law. This indicates that the returns probably are not α -stable (but it could mean as well that the returns are just not independent). Over the next few years, the stable distribution temporarily lost favor and alternative processes were suggested as mechanisms generating stock returns.

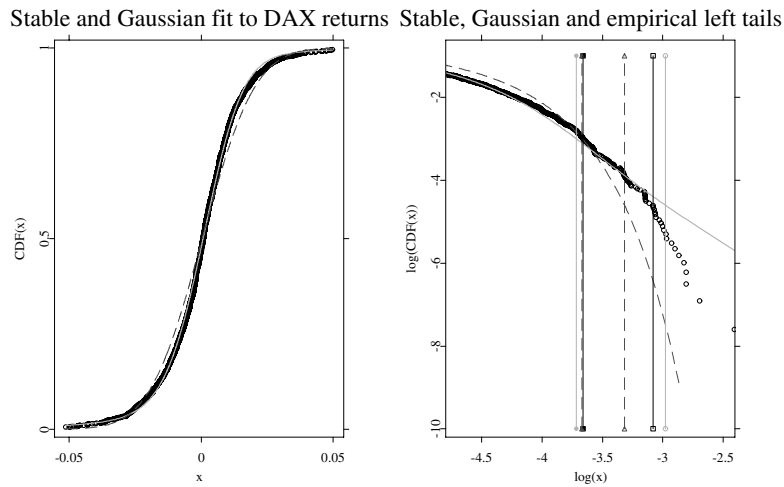


Figure 1.7. 1.7-stable (solid grey line) and Gaussian (dashed line) fits to the DAX returns (black circles) empirical cumulative distribution function from the period January 2, 1995 – December 5, 2002. For better exposition of the fit in the central part of the distribution ten largest and ten smallest returns are not illustrated in the left panel. The right panel is a magnification of the left tail fit on a double logarithmic scale. Vertical lines represent the 1.7-stable (solid grey line), Gaussian (dashed line) and empirical (solid black line) VaR estimates at the 95% (filled circles, triangles and squares) and 99% (hollow circles, triangles and squares) confidence levels. This time the stable law overestimates the tails of the empirical distribution (Q: CSAF in 06)

In mid 1990s the stable distribution hypothesis has made a dramatic comeback. Several authors have found a very good agreement of high-frequency returns with a stable distribution up to six standard deviations away from the mean (Cont, Potters and Bouchaud, 1997; Mantegna and Stanley, 1995). For more extreme observations, however, the distribution they have found falls off approximately exponentially. To cope with such observations the truncated Lévy distributions (TLD) were introduced by Mantegna and Stanley (1994). The original definition postulated a sharp truncation of the α -stable probability density function at some arbitrary point. However, later an exponential smoothing was proposed by Koponen (1995).

For $\alpha \neq 1$ the characteristic function of a symmetric TLD random variable is given by:

$$\log \phi(t) = -\frac{\sigma^\alpha}{\cos \frac{\pi\alpha}{2}} \left[(t^2 + \lambda^2)^{\alpha/2} \cos \left\{ \alpha \arctan \frac{|t|}{\lambda} \right\} - \lambda^\alpha \right],$$

where α is the tail exponent, σ is the scale parameter and λ is the truncation coefficient. Clearly the TLD reduces to the symmetric α -stable distribution ($\beta = \mu = 0$) when $\lambda = 0$. The TLD distribution exhibits the following behavior: for small and intermediate returns it behaves like a stable distribution, but for extreme returns the truncation causes the distribution to converge to a Gaussian distribution. Thus

the observation that the asset returns distribution is a TLD explains both the short-term α -stable behavior and the long run convergence to the normal distribution.

Despite these interesting features the truncated Lévy distributions have not been applied extensively to date. The most probable reason for this being the complicated definition of the TLD law. Like for α -stable distributions, only the characteristic function is known. No closed form formulas exist for the density or the distribution function. Since no integral formulas, like Zolotarev's (1986) for the α -stable laws, have been discovered as yet, statistical inference is, in general, limited to maximum likelihood utilizing the FFT technique for approximating the PDF. Moreover, compared to the stable distribution, the TLD introduces one more parameter (asymmetric TLD laws have also been considered in the literature, see e.g. Boyarchenko and Levendorskii (2000) and Koponen (1995)) making the estimation procedure even more complicated. Other parameter fitting techniques proposed so far comprise a combination of ad hoc approaches and moment matching (Boyarchenko and Levendorskii, 2000; Matacz, 2000). Better techniques have to be discovered before TLDs become a common tool in finance.

1.3 **Hyperbolic Distributions**

In response to remarkable regularities discovered by geomorphologists in the 1940s, Barndorff-Nielsen (1977) introduced the hyperbolic law for modeling the grain size distribution of windblown sand. Excellent fits were also obtained for the log-size distribution of diamonds from a large mining area in South West Africa. Almost twenty years later the hyperbolic law was found to provide a very good model for the distributions of daily stock returns from a number of leading German enterprises (Eberlein and Keller, 1995; Küchler et al., 1999), giving way to its today's use in stock price modeling (Bibby and Sørensen, 1997) and market risk measurement (Eberlein, Keller and Prause, 1998). The name of the distribution is derived from the fact that its log-density forms a hyperbola, see Fig. 1.8. Recall that the log-density of the normal distribution is a parabola. Hence the hyperbolic distribution provides the possibility of modeling heavier tails.

The hyperbolic distribution is defined as a normal variance-mean mixture where the mixing distribution is the generalized inverse Gaussian (GIG) law with parameter $\lambda = 1$, i.e. it is conditionally Gaussian, see Barndorff-Nielsen (1977) and Barndorff-Nielsen and Blaesild (1981). More precisely, a random variable Z has the hyperbolic distribution if:

$$(Z|Y) \sim N(\mu + \beta Y, Y) , \quad (1.19)$$

where Y is a generalized inverse Gaussian $GIG(\lambda = 1, \chi, \psi)$ random variable and $N(m, s^2)$ denotes the Gaussian distribution with mean m and variance s^2 . The GIG law is a very versatile positive domain three parameter distribution. It arises in the context of the first passage time of a diffusion process, when the drift and

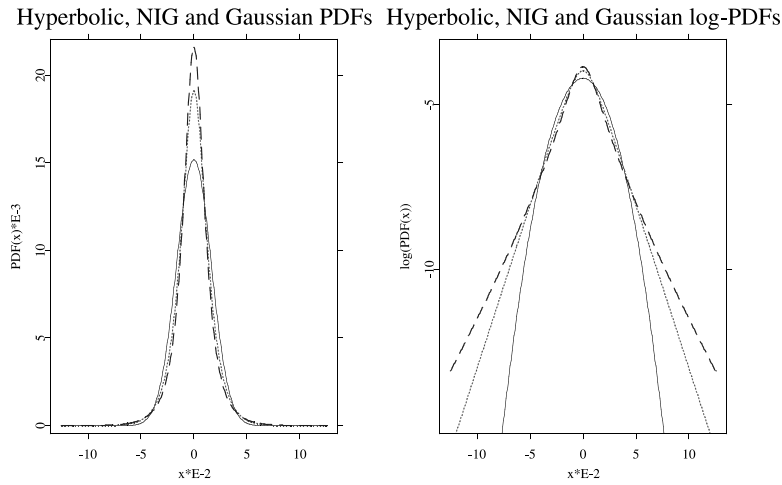


Figure 1.8. Densities and log-densities of hyperbolic (*dotted line*), NIG (*dashed line*) and Gaussian (*solid line*) distributions having the same variance, see (1.32). The name of the hyperbolic distribution is derived from the fact that its log-density forms a hyperbola, which is clearly visible in the *right panel* (Q: CSAfin07)

variance of displacement per unit time are dependent upon the current position of the particle. The probability density function of a GIG variable is given by:

$$f_{\text{GIG}}(x) = \frac{(\psi/\chi)^{\lambda/2}}{2\mathcal{K}_\lambda(\sqrt{\chi\psi})} x^{\lambda-1} e^{-\frac{1}{2}(\chi x^{-1} + \psi x)}, \quad x > 0, \quad (1.20)$$

with the parameters taking values in one of the ranges: (1) $\chi > 0, \psi \geq 0$ if $\lambda < 0$, (2) $\chi > 0, \psi > 0$ if $\lambda = 0$ or (3) $\chi \geq 0, \psi = 0$ if $\lambda > 0$. The generalized inverse Gaussian law has a number of interesting properties that we will use later in this section. The distribution of the inverse of a GIG variable is again GIG but with a different λ , namely if:

$$Y \sim \text{GIG}(\lambda, \chi, \psi) \quad \text{then} \quad Y^{-1} \sim \text{GIG}(-\lambda, \chi, \psi). \quad (1.21)$$

A GIG variable can be also reparameterized by setting $a = \sqrt{\chi/\psi}$ and $b = \sqrt{\chi\psi}$, and defining $Y = a\tilde{Y}$, where:

$$\tilde{Y} \sim \text{GIG}(\lambda, b, b). \quad (1.22)$$

The normalizing constant $\mathcal{K}_\lambda(t)$ in formula (1.20) is the modified Bessel function of the third kind with index λ , also known as the MacDonald function. It is defined as:

$$\mathcal{K}_\lambda(t) = \frac{1}{2} \int_0^\infty x^{\lambda-1} e^{-\frac{1}{2}t(x+x^{-1})} dx, \quad t > 0. \quad (1.23)$$

In the context of hyperbolic distributions, the Bessel functions are thoroughly discussed in Barndorff-Nielsen and Blaesild (1981). Here we recall only two properties that will be used later. Namely, (1) $K_\lambda(t)$ is symmetric with respect to λ , i.e. $K_\lambda(t) = K_{-\lambda}(t)$, and (2) for $\lambda = \pm\frac{1}{2}$ it can be written in a simpler form:

$$K_{\pm\frac{1}{2}}(t) = \sqrt{\frac{\pi}{2}} t^{-\frac{1}{2}} e^{-t}. \quad (1.24)$$

For other values of λ numerical approximations of the integral in (1.23) have to be used, see e.g. Campbell (1980), Press et al. (1992) or Temme (1975).

Relation (1.19) implies that a hyperbolic random variable $Z \sim H(\psi, \beta, \chi, \mu)$ can be represented in the form:

$$Z \sim \mu + \beta Y + \sqrt{Y} N(0, 1),$$

with the characteristic function:

$$\phi_Z(u) = e^{iu\mu} \int_0^\infty e^{i\beta zu - \frac{1}{2} zu^2} dF_Y(z). \quad (1.25)$$

Here $F_Y(z)$ denotes the distribution function of a generalized inverse Gaussian random variable Y with parameter $\lambda = 1$, see (1.20). Hence, the hyperbolic PDF is given by:

$$f_H(x) = \frac{\sqrt{\psi\chi}}{2\sqrt{\psi + \beta^2} K_1(\sqrt{\psi\chi})} e^{-\sqrt{\psi + \beta^2} \{\chi + (x - \mu)^2\} + \beta(x - \mu)}. \quad (1.26)$$

Sometimes another parameterization of the hyperbolic distribution with $\delta = \sqrt{\chi}$ and $\alpha = \sqrt{\psi + \beta^2}$ is used. Then the probability density function of the hyperbolic $H(\alpha, \beta, \delta, \mu)$ law can be written as:

$$f_H(x) = \frac{\sqrt{\alpha^2 - \beta^2}}{2\alpha\delta K_1(\delta\sqrt{\alpha^2 - \beta^2})} e^{-\alpha\sqrt{\delta^2 + (x - \mu)^2} + \beta(x - \mu)}, \quad (1.27)$$

where $\delta > 0$ is the scale parameter, $\mu \in \mathbb{R}$ is the location parameter and $0 \leq |\beta| < \alpha$. The latter two parameters – α and β – determine the shape, with α being responsible for the steepness and β for the skewness. In XploRe the hyperbolic density and distribution functions are implemented in the `pdfhyp` and `cdfhyp` quantlets, respectively. The calculation of the PDF is straightforward, however, the CDF has to be numerically integrated from (1.27).

The hyperbolic law is a member of a more general class of generalized hyperbolic distributions. The generalized hyperbolic law can be represented as a normal variance-mean mixture where the mixing distribution is the generalized inverse Gaussian (GIG) law with any $\lambda \in \mathbb{R}$. Hence, the generalized hyperbolic distribution

is described by five parameters $\theta = (\lambda, \alpha, \beta, \delta, \mu)$. Its probability density function is given by:

$$f_{GH}(x) = \kappa \left\{ \delta^2 + (x - \mu)^2 \right\}^{\frac{1}{2}(\lambda - \frac{1}{2})} K_{\lambda - \frac{1}{2}} \left(\alpha \sqrt{\delta^2 + (x - \mu)^2} \right) e^{\beta(x - \mu)}, \quad (1.28)$$

where:

$$\kappa = \frac{(\alpha^2 - \beta^2)^{\frac{1}{2}}}{\sqrt{2\pi} \alpha^{\lambda - \frac{1}{2}} \delta^\lambda K_\lambda \left(\delta \sqrt{\alpha^2 - \beta^2} \right)}. \quad (1.29)$$

For $|\beta + z| < \alpha$ its moment generating function takes the form:

$$M(z) = e^{\mu z} \left\{ \frac{\alpha^2 - \beta^2}{\alpha^2 - (\beta + z)^2} \right\}^{\frac{1}{2}} \frac{K_\lambda \left(\delta \sqrt{\alpha^2 - (\beta + z)^2} \right)}{K_\lambda \left(\delta \sqrt{\alpha^2 - \beta^2} \right)}. \quad (1.30)$$

Note, that $M(z)$ is smooth, i.e. infinitely many times differentiable, near 0 and hence every moment exists. If we set $\zeta = \delta \sqrt{\alpha^2 - \beta^2} = \sqrt{\psi \chi}$ then the first two moments lead to the following formulas for the mean and variance of a generalized hyperbolic random variable:

$$\mathbb{E}X = \mu + \frac{\beta \delta^2}{\zeta} \frac{K_{\lambda+1}(\zeta)}{K_\lambda(\zeta)}, \quad (1.31)$$

$$\text{Var}X = \delta^2 \left[\frac{K_{\lambda+1}(\zeta)}{\zeta K_\lambda(\zeta)} + \frac{\beta^2 \delta^2}{\zeta^2} \left\{ \frac{K_{\lambda+2}(\zeta)}{K_\lambda(\zeta)} - \left(\frac{K_{\lambda+1}(\zeta)}{\zeta K_\lambda(\zeta)} \right)^2 \right\} \right]. \quad (1.32)$$

The normal-inverse Gaussian (NIG) distributions were introduced by Barndorff-Nielsen (1995) as a subclass of the generalized hyperbolic laws obtained for $\lambda = -\frac{1}{2}$. The density of the normal-inverse Gaussian distribution is given by:

$$f_{NIG}(x) = \frac{\alpha \delta}{\pi} e^{\delta \sqrt{\alpha^2 - \beta^2} + \beta(x - \mu)} \frac{K_1 \left(\alpha \sqrt{\delta^2 + (x - \mu)^2} \right)}{\sqrt{\delta^2 + (x - \mu)^2}}. \quad (1.33)$$

In XploRe the NIG density and distribution functions are implemented in the `pdfnig` and `cdfnig` quantlets, respectively. Like for the hyperbolic distribution the calculation of the PDF is straightforward, but the CDF has to be numerically integrated from (1.33).

At the “expense” of four parameters, the NIG distribution is able to model symmetric and asymmetric distributions with possibly long tails in both directions. Its tail behavior is often classified as “semi-heavy”, i.e. the tails are lighter than those of non-Gaussian stable laws, but much heavier than Gaussian. Interestingly, if we let α tend to zero the NIG distribution converges to the Cauchy distribution (with location parameter μ and scale parameter δ), which exhibits extremely heavy tails.

The tail behavior of the NIG density is characterized by the following asymptotic relation:

$$f_{\text{NIG}}(x) \approx |x|^{-3/2} e^{(\mp\alpha+\beta)x} \quad \text{for } x \rightarrow \pm\infty. \quad (1.34)$$

In fact, this is a special case of a more general relation with the exponent of $|x|$ being equal to $\lambda - 1$ (instead of $-3/2$), which is valid for all generalized hyperbolic laws (Barndorff-Nielsen and Blaesild, 1981). Obviously, the NIG distribution may not be adequate to deal with cases of extremely heavy tails such as those of Pareto or non-Gaussian stable laws. However, empirical experience suggests an excellent fit of the NIG law to financial data (Karlis, 2002; Lillestøl, 2001; Rydberg, 1997; Venter and de Jongh, 2002). Moreover, the class of normal-inverse Gaussian distributions possesses an appealing feature that the class of hyperbolic laws does not have. Namely, it is closed under convolution, i.e. a sum of two independent NIG random variables is again NIG (Barndorff-Nielsen, 1995). In particular, if X_1 and X_2 are independent normal inverse Gaussian random variables with common parameters α and β but having different scale and location parameters $\delta_{1,2}$ and $\mu_{1,2}$, respectively, then $X = X_1 + X_2$ is $\text{NIG}(\alpha, \beta, \delta_1 + \delta_2, \mu_1 + \mu_2)$. This feature is especially useful in time scaling of risks, e.g. in deriving 10-day risks from daily risks. Only two subclasses of the generalized hyperbolic distributions are closed under convolution. The other class with this important property is the class of variance-gamma (VG) distributions, which is a limiting case obtained for $\delta \rightarrow 0$. The variance-gamma distributions (with $\beta = 0$) were introduced to the financial literature by Madan and Seneta (1990).

1.3.1 Simulation of Generalized Hyperbolic Variables

The most natural way of simulating generalized hyperbolic variables stems from the fact that they can be represented as normal variance-mean mixtures. Since the mixing distribution is the generalized inverse Gaussian law, the resulting algorithm reads as follows:

1. simulate a random variable $Y \sim \text{GIG}(\lambda, \chi, \psi) = \text{GIG}(\lambda, \delta^2, \alpha^2 - \beta^2)$;
2. simulate a standard normal random variable N , e.g. using the Box-Muller algorithm, see Sect. 1.2.3;
3. return $X = \mu + \beta Y + \sqrt{Y}N$.

The algorithm is fast and efficient if we have a handy way of simulating generalized inverse Gaussian variates. For $\lambda = -\frac{1}{2}$, i.e. when sampling from the so-called inverse Gaussian (IG) distribution, there exists an efficient procedure that utilizes a transformation yielding two roots. It starts with the observation that if we let $\vartheta = \sqrt{\chi/\psi}$ then the $\text{GIG}(-\frac{1}{2}, \chi, \psi) = \text{IG}(\chi, \psi)$ density, see (1.20), of Y can be written as:

$$f_Y(x) = \sqrt{\frac{\chi}{2\pi x^3}} \exp \left\{ \frac{-\chi(x - \vartheta)^2}{2x\vartheta^2} \right\}.$$

Now, following Shuster (1968) we may write:

$$V = \frac{\chi(Y - \vartheta)^2}{Y\vartheta^2} \sim \chi^2_{(1)}, \tag{1.35}$$

i.e. V is distributed as a chi-square random variable with one degree of freedom. As such it can be simply generated by taking a square of a standard normal random number. Unfortunately, the value of Y is not uniquely determined by (1.35). Solving this equation for Y yields two roots:

$$y_1 = \vartheta + \frac{\vartheta}{2\chi} \left(\vartheta V - \sqrt{4\vartheta\chi V + \vartheta^2 V^2} \right) \quad \text{and} \quad y_2 = \frac{\vartheta^2}{y_1}.$$

The difficulty in generating observations with the desired distribution now lies in choosing between the two roots. Michael, Schucany and Haas (1976) showed that Y can be simulated by choosing y_1 with probability $\vartheta/(\vartheta + y_1)$. So for each random observation V from a $\chi^2_{(1)}$ distribution the smaller root y_1 has to be calculated. Then an auxiliary Bernoulli trial is performed with probability $p = \vartheta/(\vartheta + y_1)$. If the trial results in a “success”, y_1 is chosen; otherwise, the larger root y_2 is selected. The `rndnig` quantlet of `XploRe`, as well as the `rnig` function of the `Rmetrics` collection of software packages for S-plus/R (see also Sect. 1.2.2 where `Rmetrics` was briefly described), utilize this routine.

In the general case, the GIG distribution – as well as the (generalized) hyperbolic law – can be simulated via the rejection algorithm. An adaptive version of this technique is used to obtain hyperbolic random numbers in the `rhypr` function of `Rmetrics`. Rejection is also implemented in the `HyperbolicDist` package for S-plus/R developed by David Scott, see the R-project home page <http://cran.r-project.org/>. The package utilizes a version of the algorithm proposed by Atkinson (1982), i.e. rejection coupled either with a two (“GIG algorithm” for any admissible value of λ) or a three part envelope (“GIGLT1 algorithm” for $0 \leq \lambda < 1$). Envelopes, also called hat or majorizing functions, provide an upper limit for the PDF of the sampled distribution. The proper choice of such functions can substantially increase the speed of computations, see Chap. II.2. As Atkinson (1982) shows, once the parameter values for these envelopes have been determined, the algorithm efficiency is reasonable for most values of the parameter space. However, finding the appropriate parameters requires optimization and makes the technique burdensome.

This difficulty led to a search for a short algorithm which would give comparable efficiencies but without the drawback of extensive numerical optimizations. A solution, based on the “ratio-of-uniforms” method, was provided a few years later by Dagpunar (1989). First, recalling properties (1.21) and (1.22), observe that we only need to find a method to simulate $\tilde{Y} \sim \text{GIG}(\lambda, b, b)$ variables and only for $\lambda \geq 0$. Next, define the relocated variable $\tilde{Y}_m = \tilde{Y} - m$, where $m = \frac{1}{b}(\lambda - 1 + \sqrt{(\lambda - 1)^2 + b^2})$ is the mode of the density of \tilde{Y} . Then the relocated variable can be generated by

taking $\tilde{Y}_m = \frac{V}{U}$, where the pair (U, V) is uniformly distributed over the region $\{(u, v) : 0 \leq u \leq \sqrt{h(\frac{v}{U})}\}$ with:

$$h(t) = (t + m)^{\lambda-1} \exp\left(-\frac{b}{2} \frac{t + m + 1}{t + m}\right), \quad \text{for } t \geq -m.$$

Since this region is irregularly shaped, it is more convenient to generate the pair (U, V) uniformly over a minimal enclosing rectangle $\{(u, v) : 0 \leq u \leq u_+, v_- \leq v \leq v_+\}$. Finally, the variate $\frac{V}{U}$ is accepted if $U^2 \leq h(\frac{V}{U})$. The efficiency of the algorithm depends on the method of deriving and the actual choice of u_+ and v_{\pm} . Further, for $\lambda \leq 1$ and $b \leq 1$ there is no need for the shift at mode m . Such a version of the algorithm is implemented in the `*gigru*` functions of UNU.RAN, a library of C functions for non-uniform random number generation developed at the Department for Statistics, Vienna University of Economics, see <http://statistik.wu-wien.ac.at/unuran/>. It is also implemented in the `gigru` function of the SSC library (a Stochastic Simulation library in C developed originally by Pierre L'Ecuyer, see <http://www.imo.umontreal.ca/~lecuyer> and Chap. II.2) and in the `rndghd` quantlet of XploRe.

1.3.2 Estimation of Parameters

Maximum Likelihood Method

The parameter estimation of generalized hyperbolic distributions can be performed by the maximum likelihood method, since there exist closed-form formulas (although, involving special functions) for the densities of these laws. The computational burden is not as heavy as for α -stable laws, but it still is considerable.

In general, the maximum likelihood estimation algorithm is as follows. For a vector of observations $x = (x_1, \dots, x_n)$, the ML estimate of the parameter vector $\theta = (\lambda, \alpha, \beta, \delta, \mu)$ is obtained by maximizing the log-likelihood function:

$$L(x; \theta) = \log \kappa + \frac{\lambda - \frac{1}{2}}{2} \sum_{i=1}^n \log(\delta^2 + (x_i - \mu)^2) + \sum_{i=1}^n \log K_{\lambda - \frac{1}{2}}\left(\alpha \sqrt{\delta^2 + (x_i - \mu)^2}\right) + \sum_{i=1}^n \beta(x_i - \mu), \quad (1.36)$$

where κ is defined by (1.29). Obviously, for hyperbolic ($\lambda = 1$) distributions the algorithm uses simpler expressions of the log-likelihood function due to relation (1.24).

The routines proposed in the literature differ in the choice of the optimization scheme. The first software product that allowed statistical inference with hyperbolic distributions – the HYP program – used a gradient search technique, see Blaesild and Sorensen (1992). In a large simulation study Prause (1999) utilized the bracketing method. The XploRe quantlets `mlehyp` and `mlehg` use yet another technique – the downhill simplex method of Nelder and Mead (1965), with slight modifications due to parameter restrictions.

The main factor for the speed of the estimation is the number of modified Bessel functions to compute. Note, that for $\lambda = 1$ (i.e. the hyperbolic distribution) this function appears only in the constant κ . For a data set with n independent observations we need to evaluate n and $n + 1$ Bessel functions for NIG and generalized hyperbolic distributions, respectively, whereas only one for the hyperbolic. This leads to a considerable reduction in the time necessary to calculate the likelihood function in the hyperbolic case. Prause (1999) reported a reduction of ca. 33%, however, the efficiency results are highly sample and implementation dependent. For example, limited simulation studies performed in XploRe revealed a 25%, 55% and 85% reduction in CPU time for samples of size 500, 1000 and 2000, respectively.

We also have to say that the optimization is challenging. Some of the parameters are hard to separate since a flat-tailed generalized hyperbolic distribution with a large scale parameter is hard to distinguish from a fat-tailed distribution with a small scale parameter, see Barndorff-Nielsen and Blaesild (1981) who observed such a behavior already for the hyperbolic law. The likelihood function with respect to these parameters then becomes very flat, and may have local minima. In the case of NIG distributions Venter and de Jongh (2002) proposed simple estimates of α and β that can be used as starting values for the ML scheme. Starting from relation (1.34) for the tails of the NIG density they derived the following approximation:

$$\alpha - \beta \sim \frac{1}{2} \frac{x_{1-f} + \mathbb{E}(X|X > x_{1-f})}{\mathbb{E}(X^2|X > x_{1-f}) - x_{1-f}\mathbb{E}(X|X > x_{1-f})},$$

$$\alpha + \beta \sim -\frac{1}{2} \frac{x_f + \mathbb{E}(X|X < x_f)}{\mathbb{E}(X^2|X < x_f) - x_f\mathbb{E}(X|X < x_f)},$$

where x_f is the f -th population quantile, see Sect. 1.2.4. After the choice of a suitable value for f , Venter and de Jongh (2002) used $f = 5\%$, the “tail estimates” of α and β are obtained by replacing the quantiles and expectations by their sample values in the above relations.

Another method of providing the starting values for the ML scheme was suggested by Prause (1999). He estimated the parameters of a symmetric ($\beta = \mu = 0$) generalized hyperbolic law with a reasonable kurtosis (i.e. with $\delta\alpha \approx 1.04$) that had the variance equal to that of the empirical distribution.

Other Methods

Besides the ML approach other estimation methods have been proposed in the literature. Prause (1999) tested different estimation techniques by replacing the log-likelihood function with other score functions, like the Anderson–Darling and Kolmogorov statistics or L^p -norms. But the results were disappointing. Lillestøl (2001) made use of the Markov chain Monte Carlo technique (see Chap. II.3), however, again the results obtained were not impressive. Karlis (2002) described an EM type algorithm (see Chap. II.5) for maximum likelihood estimation of the normal inverse Gaussian distribution. The algorithm can be programmed in any

statistical package supporting Bessel functions and it has all the properties of the standard EM algorithm, like sure, but slow, convergence, parameters in the admissible range, etc. The EM scheme can be also generalized to the family of generalized hyperbolic distributions.

1.3.3 Are Asset Returns NIG Distributed?

It is always necessary to find a reasonable tradeoff between the introduction of additional parameters and the possible improvement of the fit. Barndorff-Nielsen and Blaesild (1981) mentioned the flatness of the likelihood function for the hyperbolic distribution. The variation in the likelihood function of the generalized hyperbolic distribution is even smaller for a wide range of parameters. Consequently, the generalized hyperbolic distribution applied as a model for financial data leads to overfitting (Prause, 1999). In the empirical analysis that follows we will thus concentrate only on NIG distributions. They possess nice analytic properties and have been reported to fit financial data better than hyperbolic laws (Karlis, 2002; Lillestöl, 2001; Venter and de Jongh, 2002).

Now, we can return to the empirical analysis. This time we want to check whether DJIA and/or DAX returns can be approximated by the NIG distribution. We estimate the parameters using the maximum likelihood approach. As can be seen in Fig. 1.9 the fitted NIG distribution “misses” the very extreme DJIA returns. However, it seems to give a better fit to the central part of the empirical distribution than the α -stable law. This is confirmed by a lower value of the Kolmogorov statistics, compare Tables 1.2 and 1.4. Surprisingly, also the Anderson–Darling statistics returns a lower value, implying a better fit in the tails of the distribution as well.

Table 1.4. NIG and Gaussian fits to 2000 returns of the Dow Jones Industrial Average (DJIA) index from the period January 2, 1985 – November 30, 1992. Empirical and model based (NIG and Gaussian) VaR numbers at the 95% and 99% confidence levels are also given. The values in parentheses are the relative differences between model and empirical VaR estimates.

(Q: CSAfin08)

Parameters	α	δ or σ	β	μ
NIG fit (δ)	79.1786	0.0080	-0.3131	0.0007
Gaussian fit (σ)		0.0115		0.0006
Test values	Anderson–Darling		Kolmogorov	
NIG fit	0.3928		0.5695	
Gaussian fit	+ INF		4.5121	
VaR estimates ($\times 10^{-2}$)	95%		99%	
Empirical	1.5242		2.8922	
NIG fit	1.5194	(0.31%)	2.7855	(3.69%)
Gaussian fit	1.8350	(20.39%)	2.6191	(9.44%)

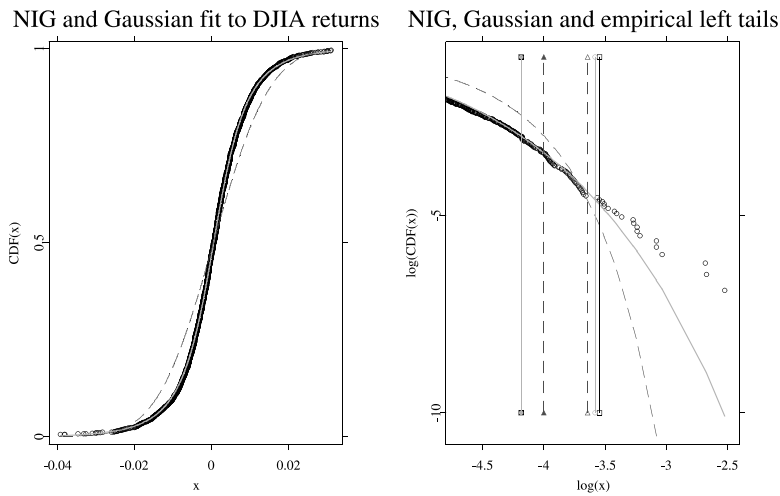


Figure 1.9. NIG (solid grey line) and Gaussian (dashed line) fits to the DJIA returns (black circles) empirical cumulative distribution function from the period January 2, 1985 – November 30, 1992. The right panel is a magnification of the left tail fit on a double logarithmic scale. Vertical lines represent the NIG (solid grey line), Gaussian (dashed line) and empirical (solid line) VaR estimates at the 95% (filled circles, triangles and squares) and 99% (hollow circles, triangles and squares) confidence levels. The NIG law slightly underfits the tails of the empirical distribution. Compare with Fig. 1.5 where the stable law is shown to fit the DJIA returns very well (Q: CSAfin08)

In the right panel of Fig. 1.9 we also plotted vertical lines representing the NIG, Gaussian and empirical daily VaR estimates at the $c = 95\%$ and 99% confidence levels. These estimates correspond to a one day VaR of a virtual portfolio consisting of one long position in the DJIA index. The NIG 95% VaR estimate matches the empirical VaR almost perfectly and the NIG 99% VaR estimate also yields a smaller difference than the stable estimate, compare Tables 1.2 and 1.4. However, if we were interested in very high confidence levels (i.e. very low quantiles) then the NIG fit would be less favorable than the stable one. Like in the stable case, no simple algorithms for inverting the NIG CDF are known but finding the right quantile could be performed through a binary search routine. For some members of the generalized hyperbolic family specialized inversion techniques have been developed. For example, Leobacher and Pillichshammer (2002) showed that the approximate inverse of the hyperbolic CDF can be computed as the solution of a first-order differential equation.

The second analyzed data set comprises 2000 returns of the Deutsche Aktienindex (DAX) index. In this case the NIG distribution offers an indisputably better fit than the Gaussian or even the α -stable law, see Table 1.5 and compare with Table 1.3. This can be also seen in Fig. 1.10. The “drop off” in the left tail of the empirical distribution is nicely caught by the NIG distribution. The empirical VaR estimates are also “caught” almost perfectly.

Table 1.5. NIG and Gaussian fits to 2000 returns of the Deutsche Aktienindex (DAX) index from the period January 2, 1995 – December 5, 2002. Empirical and model based (NIG and Gaussian) VaR numbers at the 95% and 99% confidence levels are also given. The values in parentheses are the relative differences between model and empirical VaR estimates (Q: CSAfin09)

Parameters	α	δ or σ	β	μ
NIG fit (δ)	55.4413	0.0138	-4.8692	0.0016
Gaussian fit (σ)		0.0157		0.0004
Test values	Anderson-Darling		Kolmogorov	
NIG fit	0.3604		0.8149	
Gaussian fit	16.4119		2.8197	
VaR estimates ($\times 10^{-2}$)	95%		99%	
Empirical	2.5731		4.5963	
NIG fit	2.5601	(0.51%)	4.5944	(0.04%)
Gaussian fit	2.5533	(0.77%)	3.6260	(21.11%)

1.4

Value at Risk, Portfolios and Heavy Tails

The presented examples clearly show that we not only can, but must use heavy tailed alternatives to the Gaussian law in order to obtain acceptable estimates of market losses. But can we substitute the Gaussian distribution with other distributions in Value at Risk (Expected Shortfall) calculations for whole portfolios of assets? Recall, that the definition of VaR utilizes the quantiles of the portfolio returns distribution and not the returns distribution of individual assets in the portfolio. If all asset return distributions are assumed to be Gaussian then the portfolio distribution is multivariate normal and well known statistical tools can be applied (Härdle and Simar, 2003). However, when asset returns are distributed according to a different law (or different laws!) then the multivariate distribution may be hard to tackle. In particular, linear correlation may no longer be a meaningful measure of dependence.

Luckily for us multivariate statistics offers the concept of copulas, for a review see Embrechts, Lindskog and McNeil (2003) and Nelsen (1999). In rough terms, a copula is a function $C : [0, 1]^n \rightarrow [0, 1]$ with certain special properties. Alternatively we can say that it is a multivariate distribution function defined on the unit cube $[0, 1]^n$. The technical definitions of copulas that can be found in the literature often look more complicated, but to a financial modeler, this definition is enough to build an intuition from. What is important for VaR calculations is that a copula enables us to construct a multivariate distribution function from the marginal (possibly different) distribution functions of n individual asset returns in a way that takes their dependence structure into account. This dependence structure may be no longer measured by correlation, but by other adequate functions like rank correlation, comonotonicity and, especially, tail dependence (Schmidt, 2004).

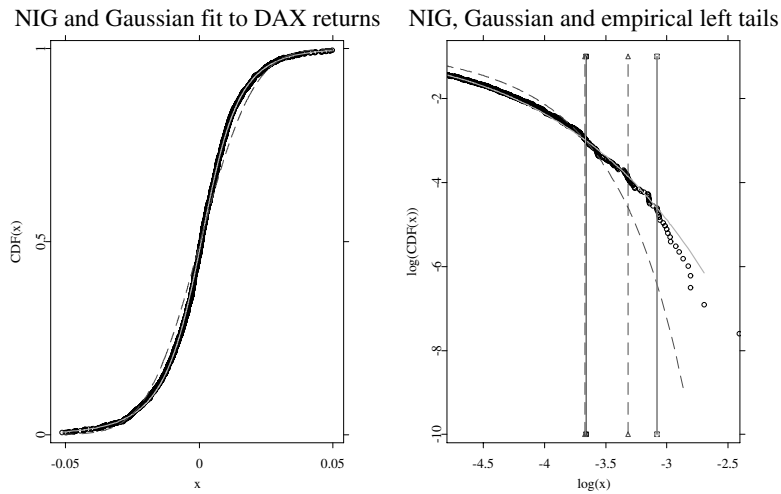


Figure 1.10. NIG (solid grey line) and Gaussian (dashed line) fits to the DAX returns (black circles) empirical cumulative distribution function from the period January 2, 1995 – December 5, 2002. The right panel is a magnification of the left tail fit on a double logarithmic scale clearly showing the superiority of the NIG distribution. Compare with Fig. 1.6 where the stable law is shown to overfit the DJIA returns. Vertical lines represent the NIG (solid grey line), Gaussian (dashed line) and empirical (solid line) VaR estimates at the 95% (filled circles, triangles and squares) and 99% (hollow circles, triangles and squares) confidence levels (Q: CSAF in09)

Moreover, it can be shown that for every multivariate distribution function there exists a copula which contains all information on dependence. For example, if the random variables are independent, then the independence copula (also known as the product copula) is just the product of n variables: $C(u_1, \dots, u_n) = u_1 \cdot \dots \cdot u_n$. If the random variables have a multivariate normal distribution with a given covariance matrix then the Gaussian copula is obtained.

Copula functions do not impose any restrictions on the model, so in order to reach a model that is to be useful in a given risk management problem, a particular specification of the copula must be chosen. From the wide variety of copulas that exist probably the elliptical and Archimedean copulas are the ones most often used in applications. Elliptical copulas are simply the copulas of elliptically contoured (or elliptical) distributions, e.g. (multivariate) normal, t , symmetric stable and symmetric generalized hyperbolic (Fang, Kotz and Ng, 1987). Rank correlation and tail dependence coefficients can be easily calculated for elliptical copulas. There are, however, drawbacks – elliptical copulas do not have closed form expressions, are restricted to have radial symmetry and have all marginal distributions of the same type. These restrictions may disqualify elliptical copulas from being used in some risk management problems. In particular, there is usually a stronger dependence between big losses (e.g. market crashes) than between big gains. Clearly, such asymmetries cannot be modeled with elliptical copulas. In contrast to elliptical copulas, all commonly encountered Archimedean copulas

have closed form expressions. Their popularity also stems from the fact that they allow for a great variety of different dependence structures (Genest and MacKay, 1986; Joe, 1997). Many interesting parametric families of copulas are Archimedean, including the well known Clayton, Frank and Gumbel copulas.

After the marginal distributions of asset returns are estimated and a particular copula type is selected, the copula parameters have to be estimated. The fit can be performed by least squares or maximum likelihood. Note, however, that for some copula types it may not be possible to maximize the likelihood function. In such cases the least squares technique should be used. A review of the estimation methods – including a description of the relevant XploRe quantlets – can be found in Rank and Siegl (2002).

For risk management purposes, we are interested in the Value at Risk of a portfolio of assets. While analytical methods for the computation of VaR exist for the multivariate normal distribution (i.e. for the Gaussian copula), in most other cases we have to use Monte Carlo simulations. A general technique for random variate generation from copulas is the conditional distributions method (Nelsen, 1999). A random vector $(u_1, \dots, u_n)^T$ having a joint distribution function C can be generated by the following algorithm:

1. simulate $u_1 \sim U(0, 1)$,
2. for $k = 2, \dots, n$ simulate $u_k \sim C_k(\cdot | u_1, \dots, u_{k-1})$.

The function $C_k(\cdot | u_1, \dots, u_{k-1})$ is the conditional distribution of the variable U_k given the values of U_1, \dots, U_{k-1} , i.e.:

$$\begin{aligned} C_k(u_k | u_1, \dots, u_{k-1}) &\stackrel{\text{def}}{=} \mathbb{P}(U_k \leq u_k | U_1 = u_1, \dots, U_{k-1} = u_{k-1}) \\ &= \frac{\partial^{k-1} c_k(u_1, \dots, u_k)}{\partial u_1 \dots \partial u_{k-1}} \bigg/ \frac{\partial^{k-1} c_{k-1}(u_1, \dots, u_{k-1})}{\partial u_1 \dots \partial u_{k-1}}, \end{aligned}$$

where c_k 's are k -dimensional margins of the n -dimensional copula C , i.e.:

$$\begin{cases} c_1(u_1) = u_1, \\ c_k(u_1, \dots, u_k) = C(u_1, \dots, u_k, 1, \dots, 1) \quad \text{for } k = 2, 3, \dots, n-1, \\ c_n(u_1, \dots, u_n) = C(u_1, \dots, u_n). \end{cases}$$

The main drawback of this method is the fact that it involves a differentiation step for each dimension of the problem. Also simulation of a $C_k(\cdot | u_1, \dots, u_{k-1})$ distributed random variable may be non-trivial. Hence, the conditional distributions technique is typically not practical in higher dimensions. For this reason, alternative methods have been developed for specific types of copulas. For example, random variables distributed according to Archimedean copula functions can be generated by the method of Marshall and Olkin (1988), which utilizes Laplace transforms. A comprehensive list of algorithms can be found in Embrechts, Lindskog and McNeil (2003). For a treatment of VaR calculations, heavy tails and copulas

consult also Bradley and Taqqu (2003), Duffie and Pan (1997) and Embrechts, McNeil and Straumann (2002).

Copulas allow us to construct models which go beyond the standard notions of correlation and multivariate Gaussian distributions. As such, in conjunction with alternative asset returns distributions discussed earlier in this chapter, they yield an ideal tool to model a wide variety of financial portfolios and products. No wonder they are gradually becoming an element of good risk management practice.

References

- Akgiray, V. and Booth, G.G. (1988). The Stable-law Model of Stock Returns, *Journal of Business & Economic Statistics* 6: 51–57.
- Artzner, P., Delbaen, F., Eber, J.-M. and Heath, D. (1999). Coherent measures of risk, *Mathematical Finance* 9: 203–228.
- Atkinson, A.C. (1982). The simulation of generalized inverse Gaussian and hyperbolic random variables, *SIAM Journal of Scientific & Statistical Computing* 3: 502–515.
- Barndorff-Nielsen, O.E. (1977). Exponentially decreasing distributions for the logarithm of particle size, *Proceedings of the Royal Society London A* 353: 401–419.
- Barndorff-Nielsen, O.E. (1995). *Normal\Inverse Gaussian Processes and the Modelling of Stock Returns*, Research Report 300, Department of Theoretical Statistics, University of Aarhus.
- Barndorff-Nielsen, O.E. and Blaesild, P. (1981). Hyperbolic distributions and ramifications: Contributions to theory and applications, in C. Taillie, G. Patil, B. Baldessari (eds.) *Statistical Distributions in Scientific Work*, Volume 4, Reidel, Dordrecht, pp. 19–44.
- Basle Committee on Banking Supervision (1995). An internal model-based approach to market risk capital requirements, <http://www.bis.org>.
- Bergström, H. (1952). On some expansions of stable distributions, *Arkiv for Matematik II*: 375–378.
- Bibby, B.M. and Sørensen, M. (1997). A hyperbolic diffusion model for stock prices, *Finance & Stochastics* 1: 25–41.
- Blaesild, P. and Sorensen, M. (1992). *HYP – a Computer Program for Analyzing Data by Means of the Hyperbolic Distribution*, Research Report 248, Department of Theoretical Statistics, Aarhus University.
- Blattberg, R.C. and Gonedes, N.J. (1974). A Comparison of the Stable and Student Distributions as Statistical Models of Stock Prices, *Journal of Business* 47: 244–280.
- Borak, Sz., Härdle, W. and Weron, R. (2004). Stable Distributions, in P. Cizek, W. Härdle, R. Weron (eds.) *Statistical Tools for Finance and Insurance*, Springer.
- Bouchaud, J.-P. and Potters, M. (2000). *Theory of Financial Risk*, Cambridge University Press, Cambridge.

- Box, G.E.P. and Muller, M.E. (1958). A note on the generation of random normal deviates, *Annals of Mathematical Statistics* **29**: 610–611.
- Boyarchenko, S.I. and Levendorskii, S.Z. (2000). Option pricing for truncated Lévy processes, *International Journal of Theoretical and Applied Finance* **3**: 549–552.
- Bradley, B.O. and Taqqu, M.S. (2003). Financial Risk and Heavy Tails, in S.T. Rachev (ed.) *Handbook of Heavy-tailed Distributions in Finance*, North Holland.
- Brorsen, B.W. and Yang, S.R. (1990). Maximum Likelihood Estimates of Symmetric Stable Distribution Parameters, *Communications in Statistics – Simulations* **19**(4): 1459–1464.
- Burnecki, K., Kukla, G., Misiorek, A. and Weron, R. (2004). Loss distributions, in P. Cizek, W. Härdle, R. Weron (eds.) *Statistical Tools for Finance and Insurance*, Springer.
- Campbell, J.B. (1980). A FORTRAN IV subroutine for the modified Bessel functions of the third kind of real order and real argument, Report NRC/ERB-925, National Research Council, Canada.
- Carr, P., Geman, H., Madan, D.B. and Yor, M. (2002). The fine structure of asset returns: an empirical investigation, *Journal of Business* **75**: 305–332.
- Chambers, J.M., Mallows, C.L. and Stuck, B.W. (1976). A Method for Simulating Stable Random Variables, *Journal of the American Statistical Association* **71**: 340–344.
- Cizek, P., Härdle, W. and Weron, R. (2004). *Statistical Tools for Finance and Insurance*, Springer. See also: <http://www.xplore-stat.de/ebooks/ebooks.html>.
- Cont, R., Potters, M. and Bouchaud, J.-P. (1997). Scaling in stock market data: Stable laws and beyond, in B. Dubrulle, F. Graner, D. Sornette (eds.) *Scale Invariance and Beyond*, Proceedings of the CNRS Workshop on Scale Invariance, Springer, Berlin.
- D’Agostino, R.B. and Stephens, M.A. (1986). *Goodness-of-Fit Techniques*, Marcel Dekker, New York.
- Dagpunar, J.S. (1989). An Easily Implemented Generalized Inverse Gaussian Generator, *Communications in Statistics – Simulations* **18**: 703–710.
- Danielsson, J., Hartmann, P. and De Vries, C.G. (1998). The cost of conservatism: Extreme returns, value at risk and the Basle multiplication factor, *Risk* **11**: 101–103.
- Dowd, K. (2002). *Measuring Market Risk*, Wiley.
- Duffie, D. and Pan, J. (1997). An overview of value at risk, *Journal of Derivatives* **4**: 7–49.
- DuMouchel, W.H. (1971). *Stable Distributions in Statistical Inference*, Ph.D. Thesis, Department of Statistics, Yale University.
- DuMouchel, W.H. (1973). On the Asymptotic Normality of the Maximum-Likelihood Estimate when Sampling from a Stable Distribution, *Annals of Statistics* **1**(5): 948–957.
- Eberlein, E. and Keller, U. (1995). Hyperbolic distributions in finance, *Bernoulli* **1**: 281–299.
- Eberlein, E., Keller, U. and Prause, K. (1998). New insights into the smile, mispricing and Value at Risk: The hyperbolic model, *Journal of Business* **71**: 371–406.

- Embrechts, P., Kluppelberg, C. and Mikosch, T. (1997). *Modelling Extremal Events for Insurance and Finance*, Springer.
- Embrechts, P., Lindskog, F. and McNeil, A.J. (2003). Modelling Dependence with Copulas and Applications to Risk Management, in S.T. Rachev (ed.) *Handbook of Heavy-tailed Distributions in Finance*, North Holland.
- Embrechts, P., McNeil, A.J. and Straumann, D. (2002). Correlation and Dependence in Risk Management: Properties and Pitfalls, in M.A.H. Dempster (ed.) *Risk Management: Value at Risk and Beyond*, Cambridge Univ. Press, Cambridge.
- Fama, E.F. (1965). The behavior of stock market prices, *Journal of Business* **38**: 34–105.
- Fama, E.F. and Roll, R. (1971). Parameter Estimates for Symmetric Stable Distributions, *Journal of the American Statistical Association* **66**: 331–338.
- Fang, K.-T., Kotz, S. and Ng, K.-W. (1987). *Symmetric Multivariate and Related Distributions*, Chapman & Hall, London.
- Fofack, H. and Nolan, J.P. (1999). Tail Behavior, Modes and Other Characteristics of Stable Distributions, *Extremes* **2**: 39–58.
- Franke, J., Härdle, W. and Stahl, G. (2000). *Measuring Risk in Complex Stochastic Systems*, Springer. See also: <http://www.xplore-stat.de/ebooks/ebooks.html>.
- Genest, C. and MacKay, J. (1986). The Joy of Copulas: Bivariate Distributions with Uniform Marginals, *The American Statistician* **40**: 280–283.
- Guillaume, D.M., Dacorogna, M.M., Dave, R.R., Müller, U.A., Olsen, R.B. and Pictet, O.V. (1997). From the birds eye to the microscope: A survey of new stylized facts of the intra-daily foreign exchange markets, *Finance & Stochastics* **1**: 95–129.
- Härdle, W., Klinke, S. and Müller, M. (2000). *XploRe Learning Guide*, Springer. See also: <http://www.xplore-stat.de/ebooks/ebooks.html>.
- Härdle, W., Kleinow, T. and Stahl, G. (2002). *Applied Quantitative Finance*, Springer. See also: <http://www.xplore-stat.de/ebooks/ebooks.html>.
- Härdle, W. and Simar, L. (2003). *Applied Multivariate Statistical Analysis*, Springer. See also: <http://www.xplore-stat.de/ebooks/ebooks.html>.
- Holt, D.R. and Crow, E.L. (1973). Tables and graphs of the stable probability density functions, *Journal of Research of the National Bureau of Standards B* **77B**: 143–198.
- Janicki, A. and Kokoszka, P. (1992). Computer investigation of the rate of convergence of LePage type series to alpha-stable random variables, *Statistica* **23**: 365–373.
- Janicki, A. and Weron, A. (1994a). Can one see α -stable variables and processes, *Statistical Science* **9**: 109–126.
- Janicki, A. and Weron, A. (1994b). *Simulation and Chaotic Behavior of α -Stable Stochastic Processes*, Marcel Dekker.
- Joe, H. (1997). *Multivariate Models and Dependence Concepts*, Chapman & Hall, London.
- Jorion, P. (2000). *Value at Risk: The New Benchmark for Managing Financial Risk*, McGraw-Hill.

- Karlis, D. (2002). An EM type algorithm for maximum likelihood estimation for the Normal Inverse Gaussian distribution, *Statistics and Probability Letters* 57: 43–52.
- Khindarova, I., Rachev, S. and Schwartz, E. (2001). Stable Modeling of Value at Risk, *Mathematical and Computer Modelling* 34: 1223–1259.
- Kogon, S.M. and Williams, D.B. (1998). Characteristic function based estimation of stable parameters, in R. Adler, R. Feldman, M. Taqqu (eds.), *A Practical Guide to Heavy Tails*, Birkhauser, pp. 311–335.
- Koponen, I. (1995). Analytic approach to the problem of convergence of truncated Levy flights towards the Gaussian stochastic process, *Physical Review E* 52: 1197–1199.
- Koutrouvelis, I.A. (1980). Regression-Type Estimation of the Parameters of Stable Laws, *Journal of the American Statistical Association* 75: 918–928.
- Küchler, U., Neumann, K., Sørensen, M. and Streller, A. (1999). Stock returns and hyperbolic distributions, *Mathematical and Computer Modelling* 29: 1–15.
- Laha, R.G. and Rohatgi, V.K. (1979). *Probability Theory*, Wiley.
- Leobacher, G. and Pillichshammer, F. (2002). A Method for Approximate Inversion of the Hyperbolic CDF, *Computing* 69: 291–303.
- LePage, R., Woodroffe, M. and Zinn, J. (1981). Convergence to a stable distribution via order statistics, *Annals of Probability* 9: 624–632.
- Lévy, P. (1925). *Calcul des Probabilités*, Gauthier Villars.
- Lillestøl, J. (2001). *Bayesian Estimation of NIG-parameters by Markov chain Monte Carlo Methods*, Discussion paper 2001/3, Department of Finance and Management Science, The Norwegian School of Economics and Business Administration.
- Madan, D.B. and Seneta, E. (1990). The variance gamma (V.G.) model for share market returns, *Journal of Business* 63: 511–524.
- Mandelbrot, B.B. (1963). The variation of certain speculative prices, *Journal of Business* 36: 394–419.
- Mantegna, R.N. (1994). Fast, accurate algorithm for numerical simulation of Levy stable stochastic processes, *Physical Review E* 49: 4677–4683.
- Mantegna, R.N. and Stanley, H.E. (1994). Stochastic processes with ultraslow convergence to a Gaussian: The truncated Lévy flight, *Physical Review Letters* 73: 2946–2949.
- Mantegna, R.N. and Stanley, H.E. (1995). Scaling behavior in the dynamics of an economic index, *Nature* 376: 46–49.
- Marshall, A.W. and Olkin, I. (1988). Families of Multivariate Distributions, *Journal of the American Statistical Association* 83: 834–841.
- Matacz, A. (2000). Financial Modeling and Option Theory with the Truncated Lévy Process, *International Journal of Theoretical and Applied Finance* 3(1): 143–160.
- McCulloch, J.H. (1986). Simple Consistent Estimators of Stable Distribution Parameters, *Communications in Statistics – Simulations* 15: 1109–1136.
- McCulloch, J.H. (1996). Financial Applications of Stable Distributions, in G.S. Maddala, C.R. Rao (eds.), *Handbook of Statistics, Vol. 14*, Elsevier, pp. 393–425.

- McCulloch, J.H. (1997). Measuring Tail Thickness to Estimate the Stable Index α : A Critique, *Journal of Business & Economic Statistics* **15**: 74–81.
- McCulloch, J.H. (1998). Numerical Approximation of the Symmetric Stable Distribution and Density, in R. Adler, R. Feldman, M. Taqqu (eds.), *A Practical Guide to Heavy Tails*, Birkhauser, pp. 489–500.
- Michael, J.R., Schucany, W.R. and Haas, R.W. (1976). Generating Random Variates Using Transformations with Multiple Roots, *The American Statistician* **30**: 88–90.
- Mittnik, S., Doganoglu, T. and Chenyao, D. (1999). Computing the Probability Density Function of the Stable Pareto Distribution, *Mathematical and Computer Modelling* **29**: 235–240.
- Mittnik, S., Rachev, S.T., Doganoglu, T. and Chenyao, D. (1999). Maximum Likelihood Estimation of Stable Pareto Models, *Mathematical and Computer Modelling* **29**: 275–293.
- Nelder, J.A. and Mead, R. (1965). A Simplex Method for Function Minimization, *The Computer Journal* **7**: 308–313.
- Nelsen, R.B. (1999). *An Introduction to Copulas*, Springer, New York.
- Nolan, J.P. (1997). Numerical Calculation of Stable Densities and Distribution Functions, *Communications in Statistics – Stochastic Models* **13**: 759–774.
- Nolan, J.P. (1999). An Algorithm for Evaluating Stable Densities in Zolotarev's (M) Parametrization, *Mathematical and Computer Modelling* **29**: 229–233.
- Nolan, J.P. (2001). Maximum Likelihood Estimation and Diagnostics for Stable Distributions, in O.E. Barndorff-Nielsen, T. Mikosch, S. Resnick (eds.), *Lévy Processes*, Birkhäuser, Boston.
- Officer, R.R. (1972). The Distribution of Stock Returns, *Journal of the American Statistical Association* **67**: 807–812.
- Ojeda, D. (2001). *Comparison of stable estimators*, Ph.D. Thesis, Department of Mathematics and Statistics, American University.
- Prause, K. (1999). *The Generalized Hyperbolic Model: Estimation, Financial Derivatives, and Risk Measures*, Ph.D. Thesis, Freiburg University, <http://www.freidok.uni-freiburg.de/volltexte/15>.
- Press, S.J. (1972). Estimation in Univariate and Multivariate Stable Distribution, *Journal of the American Statistical Association* **67**: 842–846.
- Press, W., Teukolsky, S., Vetterling, W. and Flannery, B. (1992). *Numerical Recipes in C*, Cambridge University Press. See also: <http://www.nr.com>.
- Rachev, S. and Mittnik, S. (2000). *Stable Pareto Models in Finance*, Wiley.
- Rank, J. and Siegl, T. (2002). Applications of Copulas for the Calculation of Value-at-Risk, in W. Härdle, T. Kleinow, G.P. Stahl (eds.) *Applied Quantitative Finance*, Springer.
- Rydberg, T.H. (1997). The Normal Inverse Gaussian Lévy Process: Simulation and Approximation, *Communications in Statistics – Simulations* **13(4)**: 887–910.
- Samorodnitsky, G. and Taqqu, M.S. (1994). *Stable Non-Gaussian Random Processes*, Chapman & Hall.
- Schmidt, R. (2004). Tail dependence, in P. Cizek, W. Härdle, R. Weron (eds.) *Statistical Tools for Finance and Insurance*, Springer.

- Shuster, J. (1968). On the Inverse Gaussian Distribution Function, *Journal of the American Statistical Association* **63**: 1514–1516.
- Stahl, G. (1997). Three cheers, *Risk* **10**: 67–69.
- Stute, W., Manteiga, W.G. and Quindimil, M.P. (1993). Bootstrap Based Goodness-Of-Fit-Tests, *Metrika* **40**: 243–256.
- Teichmoeller, J. (1971). A Note on the Distribution of Stock Price Changes, *Journal of the American Statistical Association* **66**: 282–284.
- Temme, N.M. (1975). On the numerical evaluation of the modified Bessel function of the third kind, *Journal of Computational Physics* **19**: 324–337.
- Venter, J.H. and de Jongh, P.J. (2002). Risk estimation using the Normal Inverse Gaussian distribution, *The Journal of Risk* **4**: 1–23.
- Weron, R. (1996). On the Chambers–Mallows–Stuck Method for Simulating Skewed Stable Random Variables, *Statistics and Probability Letters* **28**: 165–171. See also R. Weron (1996) Correction to: On the Chambers–Mallows–Stuck Method for Simulating Skewed Stable Random Variables, Research Report HSC/96/1, <http://www.im.pwr.wroc.pl/~hugo/Publications.html>.
- Weron, R. (2001). Levy–Stable Distributions Revisited: Tail Index > 2 Does Not Exclude the Levy–Stable Regime, *International Journal of Modern Physics C* **12**: 209–223.
- Zolotarev, V.M. (1964). On representation of stable laws by integrals, *Selected Translations in Mathematical Statistics and Probability* **4**: 84–88.
- Zolotarev, V.M. (1986). *One-Dimensional Stable Distributions*, American Mathematical Society.

Econometrics

IV.2

Luc Bauwens, Jeroen V.K. Rombouts

2.1	<i>Introduction</i>	952
2.2	<i>Limited Dependent Variable Models</i>	952
	Multinomial Multiperiod Probit	953
	Multivariate Probit	958
	Mixed Multinomial Logit	958
2.3	<i>Stochastic Volatility and Duration Models</i>	961
	Canonical SV Model	961
	Estimation	962
	Application	967
	Extensions of the Canonical SV Model	968
	Stochastic Duration and Intensity Models	969
2.4	<i>Finite Mixture Models</i>	971
	Inference and Identification	972
	Examples	973

2.1**Introduction**

Since the last decade we live in a digitalized world where many actions in human and economic life are monitored. This produces a continuous stream of new, rich and high quality data in the form of panels, repeated cross-sections and long time series. These data resources are available to many researchers at a low cost. This new era is fascinating for econometricians who can address many open economic questions. To do so, new models are developed that call for elaborate estimation techniques. Fast personal computers play an integral part in making it possible to deal with this increased complexity.

This chapter reviews econometric models for which statistical inference requires intensive numerical computations. A common feature of such models is that they incorporate unobserved (or latent) variables, in addition to observed ones. This often implies that the latent variables have to be integrated from the joint distribution of latent and observed variables. The implied integral is typically of high dimension and not available analytically. Simulation methods are almost always required to solve the computational issue, but they bring new problems. A general introduction on simulation based inference can be found in *Gourieroux and Monfort (1997)* and *Mariano et al. (2000)*.

The organisation of this chapter is as follows. The first section deals with limited dependent variable models, with a focus on multi-period discrete choice dynamic models. The second section treats the stochastic volatility (SV) model, used in finance and financial econometrics to calibrate the volatility of asset returns, as an alternative to the class of generalized autoregressive conditional heteroskedastic (GARCH) models. It also reviews related dynamic duration models. The last section deals with finite mixture models. Illustrative applications drawn from the recent literature are used. Programs and data are on the web site www.core.ucl.ac.be/econometrics/Bauwens/HBCS/HBCS.htm.

All the models discussed in this chapter are parametric. Nonparametric and semiparametric models may induce additional computational complexity. We refer to *Pagan and Ullah (1999)*, *Horowitz (1998)* and Chap. III.10 of this volume for examples on these methods.

2.2**Limited Dependent Variable Models**

This section deals with models in which the dependent variable is discrete. Many interesting problems like labour force participation, presidential voting, transport mode choice and brand choice are discrete in nature. In particular, we consider discrete choice models in the case where panel data are available. This allows, for example, to follow individuals with their choices over time, so that richer behavioural models can be constructed. Although the number of parameters in these models does not necessarily increase, the likelihood function, and therefore estimation, becomes more complex. In this section we describe the multinomial

multi-period probit, the multivariate probit and the mixed multinomial logit model. Examples are given.

We refer to Maddala (1983) for a general introduction to limited dependent and qualitative variables in econometrics and to Franses and Paap (2001) for a basic introduction motivating such models in relation to marketing.

Multinomial Multi-period Probit

2.2.1

Definition

Denote by U_{ijt} the unobserved utility perceived by individual i who chooses alternative j at time t . This utility may be modelled as follows

$$U_{ijt} = \mathbf{X}_{ijt}^T \boldsymbol{\beta} + \varepsilon_{ijt}, \quad (2.1)$$

where $i = 1, \dots, I$, $j = 1, \dots, J$, $t = 1, \dots, T_i$, \mathbf{X}_{ijt} is a k -dimensional vector of explanatory variables, $\boldsymbol{\beta}$ is a k -dimensional parameter vector and ε_{ijt} is a random shock known to individual i . This individual chooses alternative j in period t if

$$U_{ijt} > U_{imt} \quad \forall j \neq m. \quad (2.2)$$

We observe $\mathbf{d}_i = (d_{i1}, \dots, d_{iT_i})^T$ where $d_{it} = j$ if individual i chooses alternative j at time t . We suppose that there is always only one choice by each individual at each period, i.e. choices are mutually exclusive. The multinomial multi-period probit model is obtained by assuming

$$\boldsymbol{\varepsilon}_i = (\varepsilon_{i11}, \dots, \varepsilon_{ij1}, \dots, \varepsilon_{i1T_i}, \dots, \varepsilon_{ijT_i})^T \sim \text{IIDN}(0, \boldsymbol{\Sigma}). \quad (2.3)$$

Consequently,

$$\begin{aligned} P_i &= \text{P}(\mathbf{d}_i) = \text{P}\left(\bigcap_{m \neq d_{it}} \bigcap_{t=1}^{T_i} U_{i,d_{it},t} > U_{imt}\right) \\ &= \text{P}\left(\bigcap_{m \neq d_{it}} \bigcap_{t=1}^{T_i} \varepsilon_{i,d_{it},t} - \varepsilon_{imt} > (\mathbf{X}_{imt} - \mathbf{X}_{i,d_{it},t})^T \boldsymbol{\beta}\right), \end{aligned} \quad (2.4)$$

which is a $(T_i \times J)$ -variate integral. However, since individual choices are based on utility comparisons, it is conventional to work in utility differences relative to alternative J . If we multiply the utilities in (2.1) by a constant, we see that the probability event in (2.4) is invariant, thus a different scaling of the utilities does not alter the choices of the individuals. The rescaled relative utility is then defined as

$$\begin{aligned} \tilde{U}_{ijt} &= (U_{ijt} - U_{ijt})(\sigma_{11} + \sigma_{JJ} - 2\sigma_{1J})^{-1/2} \\ &= ((\mathbf{X}_{ijt} - \mathbf{X}_{ijt})^T \boldsymbol{\beta} + \varepsilon_{ijt} - \varepsilon_{ijt})(\sigma_{11} + \sigma_{JJ} - 2\sigma_{1J})^{-1/2} \\ &= \tilde{\mathbf{X}}_{ijt}^T \boldsymbol{\beta} + \tilde{\varepsilon}_{ijt}. \end{aligned} \quad (2.5)$$

An individual chooses alternative j in period t if

$$\tilde{U}_{ijt} > \tilde{U}_{imt} \quad \forall j \neq m. \quad (2.6)$$

As an identification restriction, one usually imposes a unit variance for the last alternative expressed in utility differences. Define

$$\tilde{\boldsymbol{\varepsilon}}_i = (\tilde{\varepsilon}_{i11}, \dots, \tilde{\varepsilon}_{i,J-1,1}, \dots, \tilde{\varepsilon}_{i1T_i}, \dots, \tilde{\varepsilon}_{i,J-1,T_i})^T \sim \text{IIDN}(0, \tilde{\boldsymbol{\Sigma}}), \quad (2.7)$$

where $\tilde{\boldsymbol{\Sigma}}$ is the transformed $\boldsymbol{\Sigma}$ with $\tilde{\sigma}_{J-1,J-1} = 1$, so that (2.4) becomes

$$P_i = \text{P} \left(\bigcap_{m \neq d_{it}} \bigcap_{t=1}^{T_i} \tilde{\varepsilon}_{i,d_{it},t} - \tilde{\varepsilon}_{imt} > (\tilde{X}_{imt} - \tilde{X}_{i,d_{it},t})^T \boldsymbol{\beta} \right), \quad (2.8)$$

which is a $T_i(J-1)$ -variate integral. Note that when the $\tilde{\varepsilon}_{ijt}$'s are serially uncorrelated, this probability event can be calculated by the product of T_i integrals of dimension $J-1$, which is easier to compute but this rules out interesting cases, see the applications below.

Estimation

This section briefly explains how the multinomial multiperiod probit model can be estimated in the classical or Bayesian framework. More details can be found in Geweke et al. (1997).

Classical Estimation. Since we assume independent observations on individuals the likelihood is

$$\text{Pr}(\mathbf{d} \mid \mathbf{X}, \boldsymbol{\beta}, \tilde{\boldsymbol{\Sigma}}) = \prod_{i=1}^I P_i, \quad (2.9)$$

where $\mathbf{d} = (d_1, \dots, d_I)$ and \mathbf{X} denotes all the observations on the explanatory variables. Evaluation of this likelihood is infeasible for reasonable values of T_i and J . Classical maximum likelihood estimation methods are usually, except in some trivial cases, based on numerical search algorithms that require many times the evaluation of the likelihood function and are therefore not suitable for this model. For more information on classical estimation, see Hajivassiliou and Ruud (1994), Gourieroux and Monfort (1997) and Hajivassiliou and Mc Fadden (1998).

Alternative estimation methods are based on simulations of the choice probabilities. The simulated maximum likelihood (SML) method maximizes the simulated likelihood which is obtained by substituting the simulated choice probabilities in (2.9). The method of simulated moments is a simulation based substitute for the generalized method of moments. For further information on these estimation methods we refer to Gourieroux and Monfort (1997).

Bayesian Inference. The posterior density is

$$\varphi(\boldsymbol{\beta}, \tilde{\boldsymbol{\Sigma}} \mid \mathbf{d}, \mathbf{X}) \propto Pr(\mathbf{d} \mid \mathbf{X}, \boldsymbol{\beta}, \tilde{\boldsymbol{\Sigma}}) \varphi(\boldsymbol{\beta}, \tilde{\boldsymbol{\Sigma}}), \tag{2.10}$$

where $\varphi(\boldsymbol{\beta}, \tilde{\boldsymbol{\Sigma}})$ is the prior density. This does not solve the problem of evaluating a high dimensional integral in the likelihood and it remains hard to compute posterior means for example. Data augmentation, see for example Tanner and Wong (1987), provides a solution because this technique allows to set up a Gibbs sampling scheme using distributions that are easy to draw from. The idea is to augment the parameter vector with $\tilde{\mathbf{U}}$, the latent utilities, so that the posterior density in (2.10) changes to

$$\varphi(\boldsymbol{\beta}, \tilde{\boldsymbol{\Sigma}}, \tilde{\mathbf{U}} \mid \mathbf{d}, \mathbf{X}) \propto Pr(\mathbf{d} \mid \mathbf{X}, \boldsymbol{\beta}, \tilde{\boldsymbol{\Sigma}}, \tilde{\mathbf{U}}) f(\tilde{\mathbf{U}} \mid \boldsymbol{\beta}, \tilde{\boldsymbol{\Sigma}}) \varphi(\boldsymbol{\beta}, \tilde{\boldsymbol{\Sigma}}) \tag{2.11}$$

implying three blocks in the Gibbs sampler: $\varphi(\boldsymbol{\beta} \mid \tilde{\boldsymbol{\Sigma}}, \tilde{\mathbf{U}}, \mathbf{d}, \mathbf{X})$, $\varphi(\tilde{\boldsymbol{\Sigma}} \mid \boldsymbol{\beta}, \tilde{\mathbf{U}}, \mathbf{d}, \mathbf{X})$ and $\varphi(\tilde{\mathbf{U}} \mid \boldsymbol{\beta}, \tilde{\boldsymbol{\Sigma}}, \mathbf{d}, \mathbf{X})$. For more details on the Gibbs sampler we refer to Chaps. II.3 and III.11. For the first two blocks, the model in (2.5) is the conventional regression model since the utilities, once simulated, are observed. For the last block, remark that $Pr(\mathbf{d} \mid \mathbf{X}, \boldsymbol{\beta}, \tilde{\boldsymbol{\Sigma}}, \tilde{\mathbf{U}})$ is an indicator function since $\tilde{\mathbf{U}}$ is consistent with \mathbf{d} or not.

Applications

It is possible to extend the model in (2.5) in various ways, such as alternative specific $\boldsymbol{\beta}$'s, individual heterogeneity or a dynamic specification.

Paap and Franses (2000) propose a dynamic specification

$$\Delta \tilde{\mathbf{U}}_{it} = \Delta \tilde{\mathbf{X}}_{it}(\boldsymbol{\alpha} + \boldsymbol{\alpha}_i) + (\boldsymbol{\Pi} - \mathbf{I}_{J-1}) (\tilde{\mathbf{U}}_{i,t-1} - \tilde{\mathbf{X}}_{i,t-1}(\boldsymbol{\beta} + \boldsymbol{\beta}_i)) + \boldsymbol{\eta}_{it}, \tag{2.12}$$

where $\tilde{\mathbf{U}}_{it}$ is the $(J - 1)$ -dimensional vector of utilities of individual i , $\Delta \tilde{\mathbf{U}}_{it} = \tilde{\mathbf{U}}_{it} - \tilde{\mathbf{U}}_{i,t-1}$, $\tilde{\mathbf{X}}_{i,t-1}$ and $\Delta \tilde{\mathbf{X}}_{it}$ are matrices of dimension $(J - 1) \times k$ for the explanatory variables, $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ are k -dimensional parameter vectors, $\boldsymbol{\Pi}$ is a $(J - 1) \times (J - 1)$ parameter matrix with eigenvalues inside the unit circle, $\boldsymbol{\eta}_{it} \sim N(0, \tilde{\boldsymbol{\Sigma}})$, and $\boldsymbol{\alpha}_i$ and $\boldsymbol{\beta}_i$ are random individual effects with the same dimension as $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$. These individual heterogeneity effects are assumed to be normally distributed: $\boldsymbol{\alpha}_i \sim N(0, \boldsymbol{\Sigma}_\alpha)$ and $\boldsymbol{\beta}_i \sim N(0, \boldsymbol{\Sigma}_\beta)$. The specification in (2.12) is a vector error-correction model where the parameters $\boldsymbol{\alpha} + \boldsymbol{\alpha}_i$ and $\boldsymbol{\beta} + \boldsymbol{\beta}_i$ measure respectively the short-run and long-run effects. The parameters in $\boldsymbol{\Pi}$ determine the speed at which deviations from the long-run relationship are adjusted.

The model parameters are $\boldsymbol{\beta}, \boldsymbol{\alpha}, \tilde{\boldsymbol{\Sigma}}, \boldsymbol{\alpha}_i, \boldsymbol{\beta}_i, \boldsymbol{\Sigma}_\beta, \boldsymbol{\Sigma}_\alpha$ and $\boldsymbol{\Pi}$ and are augmented by the latent utilities $\tilde{\mathbf{U}}_{it}$. Bayesian inference may be done by Gibbs sampling as described in the estimation part above. Table 2.1 describes for each of the nine blocks which posterior distribution is used. For example, $\boldsymbol{\beta}$ has a conditional (on all other parameters) posterior density that is normal.

As an illustration we reproduce the results of Paap and Franses (2000), who provided their Gauss code (which we slightly modified). They use optical scanner data on purchases of four brands of saltine crackers. Chintagunta and Honore (1996) use the same data set to estimate a static multinomial probit model. The data set

Table 2.1. Summary of conditional posteriors for (2.12)

Parameter	Conditional posterior
$\beta, \beta_i, \alpha, \alpha_i$	Multivariate normal distributions
$\tilde{\Sigma}, \Sigma_\alpha, \Sigma_\beta$	Inverted Wishart distributions
Π	Matrix normal distribution
\tilde{U}_{it}	Truncated multivariate normal

contains all purchases (choices) of crackers of 136 households over a period of two years, yielding 3292 observations. Variables such as prices of the brands and whether there was a display and/or newspaper feature of the considered brands at the time of purchase are also observed and used as the explanatory variables forming X_{ijt} (and then transformed into \tilde{X}_{ijt}). Table 2.2 gives the means of these variables. Display and Feature are dummy variables, e.g. Sunshine was displayed 13% and was featured 4% of the purchase occasions. The average market shares reflect the observed individual choices, with e.g. 7% of the choices on Sunshine.

Table 2.2. Means of X_{it} variables in (2.12)

	Sunshine	Keebler	Nabisco	Private Label
Market share	0.07	0.07	0.54	0.32
Display	0.13	0.11	0.34	0.10
Feature	0.04	0.04	0.09	0.05
Price	0.96	1.13	1.08	0.68

Table 2.3 shows posterior means and standard deviations for the α and β parameters. They are computed from 50,000 draws after dropping 20,000 initial draws. The prior on $\tilde{\Sigma}$ is inverted Wishart, denoted by $IW(S, \nu)$, with $\nu = 10$ and S chosen such that $E(\tilde{\Sigma}) = I_3$. Note that Paap and Franses (2000) use a prior such that $E(\tilde{\Sigma}^{-1}) = I_3$. For the other parameters we put uninformative priors. As expected, Display and Feature have positive effects on the choice probabilities and price has a negative effect. This holds both in the short run and the long run. With respect to the private label (which serves as reference category), the posterior means of the intercepts are positive except for the first label whose intercept is imprecisely estimated.

Table 2.3. Posterior moments of β and α in (2.12)

	β parameter		α parameter		Intercepts	
	mean	st. dev.	mean	st. dev.	mean	st. dev.
Display	0.307	(0.136)	0.102	(0.076)	Sunshine	-0.071 (0.253)
Feature	0.353	(0.244)	0.234	(0.090)	Keebler	0.512 (0.212)
Price	-1.711	(0.426)	-2.226	(0.344)	Nabisco	1.579 (0.354)

Table 2.4 gives the posterior means and standard deviations of $\tilde{\Sigma}$, Π , $\tilde{\Sigma}_\beta$ and $\tilde{\Sigma}_\alpha$. Note that the reported last element of $\tilde{\Sigma}$ is equal to 1 in order to identify the model. This is done, after running the Gibbs sampler with $\tilde{\Sigma}$ unrestricted, by dividing the variance related parameter draws by $\tilde{\sigma}_{j-1,j-1}$. The other parameter draws are divided by the square root of the same quantity. McCulloch et al. (2000) propose an alternative approach where $\tilde{\Sigma}_{j-1,j-1}$ is fixed to 1 by construction, i.e. a fully identified parameter approach. They write

$$\tilde{\Sigma} = \begin{pmatrix} \Phi + \gamma\gamma^T & \gamma \\ \gamma^T & 1 \end{pmatrix} \tag{2.13}$$

and show that the conditional posterior of γ is normal and that of Φ is Wishart, so that draws of $\tilde{\Sigma}$ are easily obtained. This approach is of particular interest when a sufficiently informative prior on $\tilde{\Sigma}$ is used. A drawback of this approach is that the Gibbs sampler has higher autocorrelation and that it is more sensitive to initial conditions.

The relatively large posterior means of the diagonal elements of Π show that there is persistence in brand choice. The matrices $\tilde{\Sigma}_\beta$ and $\tilde{\Sigma}_\alpha$ measure the unobserved heterogeneity. There seems to be substantial heterogeneity across the

Table 2.4. Posterior means and standard deviations of $\tilde{\Sigma}$, Π , $\tilde{\Sigma}_\beta$ and $\tilde{\Sigma}_\alpha$ in (2.12)

$\tilde{\Sigma}$ $\begin{pmatrix} 0.563 & -0.102 & 0.433 \\ (0.179) & (0.096) & (0.087) \\ & 0.241 & 0.293 \\ & (0.119) & (0.069) \\ & & 1 \end{pmatrix}$	Π $\begin{pmatrix} 0.474 & 0.213 & 0.054 \\ (0.103) & (0.134) & (0.066) \\ 0.440 & 0.685 & -0.196 \\ (0.067) & (0.081) & (0.049) \\ -0.099 & -0.161 & 0.421 \\ (0.091) & (0.138) & (0.087) \end{pmatrix}$
$\tilde{\Sigma}_\beta$ $\begin{pmatrix} 0.431 & -0.267 & 0.335 & -0.176 & -0.100 & 0.087 \\ (0.201) & (0.250) & (0.463) & (0.247) & (0.209) & (0.401) \\ & 1.053 & 0.281 & 0.412 & 0.306 & 0.721 \\ & (0.603) & (0.774) & (0.352) & (0.372) & (0.719) \\ & & 5.445 & -1.310 & -1.010 & 0.539 \\ & & (2.268) & (0.999) & (0.853) & (1.120) \\ & & & 1.919 & 1.225 & 1.950 \\ & & & (0.672) & (0.560) & (0.664) \\ & & & & 1.496 & 1.564 \\ & & & & (0.879) & (0.816) \\ & & & & & 4.915 \\ & & & & & (1.319) \end{pmatrix}$	$\tilde{\Sigma}_\alpha$ $\begin{pmatrix} 0.207 & -0.023 & -0.004 \\ (0.091) & (0.075) & (0.220) \\ & 0.382 & 0.217 \\ & (0.144) & (0.366) \\ & & 6.672 \\ & & (2.453) \end{pmatrix}$

individuals, especially for the price of the products (see the third diagonal elements of both matrices). The last three elements in $\tilde{\Sigma}_\beta$ are related to the intercepts.

The multinomial probit model is frequently used for marketing purposes. For example, Allenby and Rossi (1999) use ketchup purchase data to emphasize the importance of a detailed understanding of the distribution of consumer heterogeneity and identification of preferences at the customer level. In fact, the disaggregate nature of many marketing decisions creates the need for models of consumer heterogeneity which pool data across individuals while allowing for the analysis of individual model parameters. The Bayesian approach is particularly suited for that, contrary to classical approaches that yields only aggregate summaries of heterogeneity.

2.2.2 Multivariate Probit

The multivariate probit model relaxes the assumption that choices are mutually exclusive, as in the multinomial model discussed before. In that case, d_i may contain several 1's. Chib and Greenberg (1998) discuss classical and Bayesian inference for this model. They also provide examples on voting behavior, on health effects of air pollution and on labour force participation.

2.2.3 Mixed Multinomial Logit

Definition

The multinomial logit model is defined as in (2.1), except that the random shock ε_{ijt} is extreme value (or Gumbel) distributed. This gives rise to the independence from irrelevant alternatives (IIA) property which essentially means that $\text{Cov}(U_{ijt}, U_{ikt}) = 0 \forall j, \forall k$. Like the probit model, the mixed multinomial logit (MMNL) model alleviates this restrictive IIA property by treating the β parameter as a random vector with density $f_\theta(\beta)$. The latter density is called the mixing density and is usually assumed to be a normal, lognormal, triangular or uniform distribution. To make clear why this model does not suffer from the IIA property, consider the following example. Suppose that there is only explanatory variable and that $\beta \sim N(\bar{\beta}, \bar{\sigma}^2)$. We can then write (2.1) as

$$\begin{aligned} U_{ijt} &= X_{ijt}\bar{\beta} + X_{ijt}\bar{\sigma}z + \varepsilon_{ijt} \\ &= X_{ijt}\bar{\beta} + \varepsilon_{ijt}^* , \end{aligned} \quad (2.14)$$

where $z \sim N(0, 1)$, implying that the variance of ε_{ijt}^* depends on the explanatory variable and that there is nonzero covariance between utilities for different alternatives.

The mixed logit probability is given by

$$P_i = \int \prod_{t=1}^{T_i} \left(\frac{e^{X_{ijt}^T \beta}}{\sum_{j=1}^J e^{X_{ijt}^T \beta}} \right) f_\theta(\beta) d\beta , \quad (2.15)$$

where the term between brackets is the logistic distribution arising from the difference between two extreme value distributions. The model parameter is θ . Note that one may want to keep elements of β fixed as in the usual logit model. One usually keeps random the elements of β corresponding to the variables that are believed to create correlation between alternatives. The mixed logit model is quite general. McFadden and Train (2000) demonstrate that any random utility model can be approximated to any degree of accuracy by a mixed logit with appropriate choice of variables and mixing distribution.

Estimation

Classical Estimation. Estimation of the MMNL model can be done by SML or the method of simulated moments or simulated scores. To do this, the logit probability in (2.15) is replaced by its simulated counterpart

$$SP_i = \frac{1}{R} \sum_{r=1}^R \prod_{t=1}^{T_i} \left(\frac{e^{X_{ijt}^T \beta^r}}{\sum_{j=1}^J e^{X_{ijt}^T \beta^r}} \right), \quad (2.16)$$

where the $\{\beta^r\}_{r=1}^R$ are i.i.d. draws of $f_{\theta}(\beta)$. The simulated likelihood is the product of all the individual SP_i 's. The simulated log-likelihood can be maximized with respect to θ using numerical optimization techniques like the Newton–Raphson algorithm. To avoid an erratic behaviour of the simulated objective function for different values of θ , the same sequences of basic random numbers is used to generate the sequence $\{\beta^r\}$ used during all the iterations of the optimizer (this is referred to as the technique of ‘common random numbers’).

According to Gouriéroux and Monfort (1997) the SML estimator is asymptotically equivalent to the ML estimator if T (the total number of observations) and R both tend to infinity and $\sqrt{T}/R \rightarrow 0$. In practice, it is sufficient to fix R at a moderate value.

The approximation of an integral like in (2.15) by the use of pseudo-random numbers may be questioned. Bhat (2001) implements an alternative quasi-random SML method which uses quasi-random numbers. Like pseudo-random sequences, quasi-random sequences, such as Halton sequences, are deterministic, but they are more uniformly distributed in the domain of integration than pseudo-random ones. The numerical experiments indicate that the quasi-random method provides considerably better accuracy with much fewer draws and computational time than does the usual random method.

Bayesian Inference. Let us suppose that the mixing distribution is Gaussian, that is, the vector β is normally distributed with mean \mathbf{b} and variance matrix \mathbf{W} . The posterior density for I individuals can be written as

$$\varphi(\mathbf{b}, \mathbf{W} \mid \mathbf{d}, \mathbf{X}) \propto Pr(\mathbf{d} \mid \mathbf{X}, \mathbf{b}, \mathbf{W}) \varphi(\mathbf{b}, \mathbf{W}), \quad (2.17)$$

where $Pr(\mathbf{d} \mid \mathbf{X}, \mathbf{b}, \mathbf{W}) = \prod_{i=1}^I P_i$ and $\varphi(\mathbf{b}, \mathbf{W})$ is the prior density on \mathbf{b} and \mathbf{W} . Sampling from (2.17) is difficult because P_i is an integral without a closed form as discussed above. We would like to condition on $\boldsymbol{\beta}$ such that the choice probabilities are easy to calculate. For this purpose we augment the model parameter vector with $\boldsymbol{\beta}$. It is convenient to write $\boldsymbol{\beta}_i$ instead of $\boldsymbol{\beta}$ to interpret the random coefficients as representing heterogeneity among individuals. The $\boldsymbol{\beta}_i$'s are independent and identically distributed with mixing distribution $f(\cdot \mid \mathbf{b}, \mathbf{W})$. The posterior can then be written as

$$\varphi(\mathbf{b}, \mathbf{W}, \boldsymbol{\beta}_I \mid \mathbf{d}, \mathbf{X}) \propto Pr(\mathbf{d} \mid \mathbf{X}, \boldsymbol{\beta}_I) f(\boldsymbol{\beta}_I \mid \mathbf{b}, \mathbf{W}) \varphi(\mathbf{b}, \mathbf{W}), \quad (2.18)$$

where $\boldsymbol{\beta}_I$ collects the $\boldsymbol{\beta}_i$'s for all the I individuals. Draws from this posterior density can be obtained by using the Gibbs sampler. Table 2.5 summarizes the three blocks of the sampler.

Table 2.5. Summary of conditional posteriors for MMNL model

Parameter	Conditional posterior or sampling method
\mathbf{b}	Multivariate normal distribution
\mathbf{W}	Inverted Wishart distribution
$\boldsymbol{\beta}_I$	Metropolis Hastings algorithm

For the first two blocks the conditional posterior densities are known and are easy to sample from. The last block is more difficult. To sample from this density, a Metropolis Hastings (MH) algorithm is set up. Note that only one iteration is necessary such that simulation within the Gibbs sampler is avoided. See Train (2003), Chap. 12, for a detailed description of the MH algorithm for the mixed logit model and for guidelines about how to deal with other mixing densities. More general information on the MH algorithm can be found in Chap. II.3.

Bayesian inference in the mixed logit model is called hierarchical Bayes because of the hierarchy of parameters. At the first level, there are the individual parameters $\boldsymbol{\beta}_i$ which are distributed with mean $\boldsymbol{\beta}$ and variance matrix \mathbf{W} . The latter are called hyper-parameters, on which we have also prior densities. They form the second level of the hierarchy.

Application

We reproduce the results of McFadden and Train (2000) using their Gauss code available on the web site elsa.berkeley.edu/~train/software.html. They analyse the demand for alternative vehicles. There are 4654 respondents who choose among six alternatives (two alternatives run on electricity only). There are 21 explanatory variables among which 4 are considered to have a random effect. The mixing distributions for these random coefficients are independent normal distributions. The model is estimated by SML and uses $R = 250$ replications per observation. Table 2.6 includes partly the estimation results of the MMNL model. We report the estimates and standard errors of the parameters of the normal mixing distributions, but we

do not report the estimates of the fixed effect parameters corresponding to the 17 other explanatory variables. For example, the luggage space error component induces greater covariance in the stochastic part of utility for pairs of vehicles with greater luggage space. We refer to McFadden and Train (2000) or Brownstone and Train (1999) for more interpretations of the results.

Train (2003) provides more information and pedagogical examples on the mixed multinomial model.

Table 2.6. SML estimates of MMNL random effect parameters

Variable	Mean		Standard deviation	
Electric vehicle (EV) dummy	-1.032	(0.503)	2.466	(0.720)
Compressed natural gas (CNG) dummy	0.626	(0.167)	1.072	(0.411)
Size	1.435	(0.499)	7.457	(2.043)
Luggage space	1.702	(0.586)	5.998	(1.664)

Robust standard errors within parentheses

Stochastic Volatility and Duration Models

2.3

Stochastic volatility (SV) models may be used as an alternative to generalized autoregressive conditional heteroskedastic (GARCH) models as a way to model the time-varying volatility of asset returns. Time series of asset returns feature stylized facts, the most important being volatility clustering, which produces a slowly decreasing positive autocorrelation function of the squared returns, starting at a low value (about 0.15). Another stylized fact is excess kurtosis of the distribution (with respect to the Gaussian distribution). See Bollerslev et al. (1994) for a detailed list of the stylized facts and a survey of GARCH models, Shephard (1996) for a comparative survey of GARCH and SV models, and Ghysels et al. (1996) for a survey of SV models focused on their theoretical foundations and their applications in finance. The first four parts of this section deal with SV models while in Sect. 2.3.5 we survey similar models for dynamic duration analysis.

Canonical SV Model

2.3.1

The simplest version of a SV model is given by

$$\begin{aligned}
 y_t &= \exp(h_t/2) u_t, & u_t &\sim N(0, 1), & t &= 1, \dots, n, \\
 h_t &= \omega + \beta h_{t-1} + \sigma v_t, & v_t &\sim N(0, 1),
 \end{aligned}
 \tag{2.19}$$

where y_t is a return measured at t , h_t is the unobserved log-volatility of y_t , $\{u_t\}$ and $\{v_t\}$ are mutually independent sequences, (ω, β, σ) are parameters to be estimated, jointly denoted θ . The parameter space is $\mathbb{R} \times (-1, 1) \times \mathbb{R}_+$. The restriction on β ensures the strict stationarity of y_t . Estimates of β are typically quite close

to 1 (in agreement with the first stylized fact), thus β is a ‘persistence’ parameter of the volatility. The unconditional mean of h_t is $\mu = \omega/(1 - \beta)$ and the second equation may be parametrized using μ by writing $h_t = \mu + \beta(h_{t-1} - \mu) + \sigma v_t$. Another parametrization removes ω from the second equation while writing the first as $y_t = \tau \exp(h_t/2) u_t$ where $\tau = \exp(\omega/2)$. These different parametrizations are in one-to-one correspondance. Which one to choose is mainly a matter of convenience and numerical efficiency of estimation algorithms.

For further use, let y and h denote the $n \times 1$ vectors of observed returns and unobserved log-volatilities, respectively.

2.3.2 Estimation

Estimation of the parameters of the canonical SV model may be done by the maximum likelihood (ML) method or by Bayesian inference. Other methods have been used but they will not be considered here. We refer to Ghysels et al. (1996), Sect. 5, for a review. ML and, in principle, Bayesian estimation require to compute the likelihood function of an observed sample, which is a difficult task. Indeed, the density of y given θ and an initial condition h_0 (not explicitly written in the following expressions) requires to compute a multiple integral which has a dimension equal to the sample size:

$$f(y|\theta) = \int f(y, h|\theta) dh \tag{2.20}$$

$$= \int f(y|h, \theta) f(h|\theta) dh \tag{2.21}$$

$$= \int \prod_{t=1}^n f(y_t, h_t | Y_{t-1}, H_{t-1}, \theta) dh, \tag{2.22}$$

where $Y_t = \{y_i\}_{i=1}^t$ and $H_t = \{h_i\}_{i=0}^t$. For model (2.19), this is

$$\int \prod_{t=1}^n f_N(y_t | 0, e^{h_t}) f_N(h_t | \omega + \beta h_{t-1}, \sigma^2) dh, \tag{2.23}$$

where $f_N(x|\mu, \sigma^2)$ denotes the normal density function of x , with parameters μ and σ^2 . An analytical solution to the integration problem is not available. Even a term by term numerical approximation by a quadrature rule is precluded: the integral of $N(0, \exp(h_n)) \times N(\omega + \beta h_{n-1}, \sigma^2)$ with respect to h_n depends on h_{n-1} , and has to be carried over in the previous product, and so on until h_1 . This would result in an explosion of the number of function evaluations. Simulation methods are therefore used.

Two methods directly approximate (2.22): efficient importance sampling (EIS), and Monte Carlo maximum likelihood (MCML). Another approach, which can only be used for Bayesian inference, works with $f(y, h|\theta)$ as data density, and produces a posterior joint density for θ, h given y . The posterior density is simulated

by a Monte Carlo Markov chain (MCMC) algorithm, which produces simulated values of θ and \mathbf{h} . Posterior moments and marginal densities of θ are then estimated by their simulated sample counterparts. We pursue by describing each method.

EIS (Liesenfeld and Richard (2003))

A look at (2.23) suggests to sample R sequences $\{h_t^r \sim N(\omega + \beta h_{t-1}, \sigma^2)\}_{t=1}^n, r = 1 \dots R$, and to approximate it by $(1/R) \sum_{r=1}^R \prod_{t=1}^n N(0, \exp(h_t^r))$. This direct method proves to be inefficient. Intuitively, the sampled sequences of h_t are not linked to the observations y_t . To improve upon this, the integral (2.22), which is the convenient expression to present EIS, is expressed as

$$\int \prod_{t=1}^n \frac{f(y_t, h_t | \mathbf{Y}_{t-1}, \mathbf{H}_{t-1}, \theta)}{m(h_t | \mathbf{H}_{t-1}, \phi_t)} m(h_t | \mathbf{H}_{t-1}, \phi_t) d\mathbf{h}, \tag{2.24}$$

where $\{m(h_t | \mathbf{H}_{t-1}, \phi_t)\}_{t=1}^n$ is a sequence of importance density functions, indexed by parameters $\{\phi_t\}$. These importance functions serve to generate R random draws $\{h_t^1, h_t^2 \dots h_t^R\}_{t=1}^n$, such that the integral is approximated by the sample mean

$$\frac{1}{R} \sum_{r=1}^R \prod_{t=1}^n \frac{f(y_t, h_t^r | \mathbf{Y}_{t-1}, \mathbf{H}_{t-1}^r, \theta)}{m(h_t^r | \mathbf{H}_{t-1}^r, \phi_t)}. \tag{2.25}$$

The essential point is to choose the form of $m(\cdot)$ and its auxiliary parameters ϕ_t so as to secure a good match between the product of $m(h_t | \mathbf{H}_{t-1}, \phi_t)$ and the product of $f(y_t, h_t | \mathbf{Y}_{t-1}, \mathbf{H}_{t-1}, \theta)$ viewed as a function of \mathbf{h} . A relevant good match criterion is provided by a choice of $\{\phi_t\}$, for a given family of densities for $m(\cdot)$, based on the minimization of the Monte Carlo variance of the mean (2.25). The choice of $\{\phi_t\}$ is explained below, after the choice of $m(\cdot)$.

A convenient choice for $m(\cdot)$ is the Gaussian family of distributions. A Gaussian approximation to $f(\cdot)$, as a function of h_t , given y_t and h_{t-1} , turns out to be efficient. It can be expressed as proportional to $\exp(\phi_{1,t} h_t + \phi_{2,t} h_t^2)$, where $(\phi_{1,t}, \phi_{2,t}) = \phi_t$, the auxiliary parameters. It is convenient to multiply it with $\exp[-0.5\sigma^{-2}(-2m_t h_t + h_t^2 + m_t^2)]$, where $m_t = \omega + \beta h_{t-1}$, which comes from the $N(m_t, \sigma^2)$ term included in $f(y_t, h_t | \mathbf{Y}_{t-1}, \mathbf{H}_{t-1}, \theta)$. The product of these two exponential functions can be expressed as a Gaussian density $N(\mu_t, \sigma_t^2)$, where

$$\mu_t = \sigma_t^2(m_t/\sigma^2 + \phi_{1,t}), \quad \sigma_t^2 = \sigma^2/(1 - 2\sigma^2\phi_{2,t}). \tag{2.26}$$

The choice of the auxiliary parameters can be split into n separate problems, one for each t . It amounts to minimize the sum of squared deviations between $\ln f(y_t | \mathbf{Y}_{t-1}, \mathbf{H}_t^r, \theta)$ plus a correction term, see (2.27), and $\phi_{0,t} + \phi_{1,t} h_t^r + \phi_{2,t} (h_t^r)^2$ where $\phi_{0,t}$ is an auxiliary intercept term. This problem is easily solved by ordinary least squares. See Liesenfeld and Richard (2003) for a detailed explanation.

Let us summarize the core of the EIS algorithm in three steps (for given θ and y):
Step 1: Generate R trajectories $\{h_t^r\}$ using the ‘natural’ samplers $\{N(m_t, \sigma^2)\}$.
Step 2: For each t (starting from $t = n$ and ending at $t = 1$), using the R observations generated in the previous step, estimate by OLS the regression

$$-\frac{1}{2} \left[h_t^r + y_t^2 e^{-h_t^r} + \left(\frac{m_{t+1}^r}{\sigma_{t+1}^r} \right)^2 - \left(\frac{m_{t+1}^r}{\sigma} \right)^2 \right] = \phi_{0,t} + \phi_{1,t} h_t^r + \phi_{2,t} (h_t^r)^2 + \varepsilon_t^r, \quad (2.27)$$

where ε_t^r is an error term. For $t = n$, the dependent variable does not include the last two terms in the square brackets. The superscript r on μ_{t+1} , σ_{t+1} and m_{t+1} indicates that these quantities are evaluated using the r -th trajectory.

Step 3: Generate R trajectories $\{h_t^r\}$ using the efficient samplers $\{N(\mu_t, \sigma_t^2)\}$ and finally compute (2.25).

Steps 1 to 3 should be iterated about five times to improve the efficiency of the approximations. This is done by replacing the natural sampler in Step 1 by the importance functions built in the previous iteration. It is also possible to start Step 1 of the first iteration with a more efficient sampler than the natural one. This is achieved by multiplying the natural sampler by a normal approximation to $f(y_t | h_t, h_{t-1}, \theta) \propto \exp\{-0.5[y_t^2 \exp(-h_t) + h_t]\}$. The normal approximation is based on a second-order Taylor series expansion of the argument of the exponential in the previous expression around $h_t = 0$. In this way, the initial importance sampler links y_t and h_t . This enables one to reduce to three (instead of five) the number of iterations over the three steps. In practical implementations, R can be fixed to 50. When computing (2.25) for different values of θ , such as in a numerical optimizer, it is important to use common random numbers to generate the set of R trajectories $\{h_t^r\}$ that serve in the computations.

It is also easy to compute by EIS filtered estimates of functions of h_t , such as the conditional standard deviation $\exp(h_t/2)$, conditional on the past returns (but not on the lagged unobserved h_t), given a value of θ (such as the ML estimate). Diagnostics on the model specification are then obtained as a byproduct: if the model is correctly specified, y_t divided by the filtered estimates of $\exp(h_t/2)$ is a residual that has zero mean, unit variance, and is serially uncorrelated (this also holds for the squared residual).

Richard (1998) contains a general presentation of EIS and its properties.

MCML (Durbin and Koopman (1997))

The likelihood to be computed at y (the data) and any given θ is equal to $f(y|\theta)$ and is conveniently expressed as (2.21) for this method. This quantity is approximated by importance sampling with an importance function defined from an approximating model. The latter is obtained by using the state space representation of the canonical SV model (parametrized with τ):

$$\ln y_t^2 = \ln \tau^2 + h_t + \varepsilon_t, \quad (2.28)$$

$$h_t = \beta h_{t-1} + \sigma v_t. \quad (2.29)$$

In the canonical SV model, $\varepsilon_t = \ln u_t^2$ is distributed as the logarithm of a $\chi^2(1)$ random variable. However the approximating model replaces this with a Gaussian distribution (defined below), keeping the state equation unchanged. Therefore, the whole machinery of the Kalman filter is applicable to the approximating model, which is a Gaussian linear state space model. If we denote by $g(\mathbf{h}|\mathbf{y}, \boldsymbol{\theta})$ the importance function that serves to simulate \mathbf{h} (see below), we have

$$f(\mathbf{y}|\boldsymbol{\theta}) = \int \frac{f(\mathbf{y}|\mathbf{h}, \boldsymbol{\theta})f(\mathbf{h}|\boldsymbol{\theta})}{g(\mathbf{h}|\mathbf{y}, \boldsymbol{\theta})} g(\mathbf{h}|\mathbf{y}, \boldsymbol{\theta}) d\mathbf{h} \quad (2.30)$$

$$= g(\mathbf{y}|\boldsymbol{\theta}) \int \frac{f(\mathbf{y}|\mathbf{h}, \boldsymbol{\theta})}{g(\mathbf{y}|\mathbf{h}, \boldsymbol{\theta})} g(\mathbf{h}|\mathbf{y}, \boldsymbol{\theta}) d\mathbf{h}, \quad (2.31)$$

where the second equality results from $g(\mathbf{h}|\mathbf{y}, \boldsymbol{\theta})g(\mathbf{y}|\boldsymbol{\theta}) = g(\mathbf{y}|\mathbf{h}, \boldsymbol{\theta})g(\mathbf{h}|\boldsymbol{\theta})$ and $g(\mathbf{h}|\boldsymbol{\theta}) = f(\mathbf{h}|\boldsymbol{\theta})$. All the densities $g(\cdot)$ and $g(\cdot|\cdot)$ are defined from the approximating Gaussian model. In particular, $g(\mathbf{y}|\boldsymbol{\theta})$ is the likelihood function of the Gaussian linear state space model and is easy to compute by the Kalman filter (see the appendix to Sandman and Koopman (1998) for all details). Likewise, $g(\mathbf{y}|\mathbf{h}, \boldsymbol{\theta})$ obtains from the Gaussian densities $g(\ln y_t^2|h_t, \boldsymbol{\theta})$ resulting from (2.28) with $\varepsilon_t \sim N(a_t, s_t^2)$ where a_t and s_t^2 are chosen so that $g(\mathbf{y}|\mathbf{h}, \boldsymbol{\theta})$ is as close as possible to $f(\mathbf{y}|\mathbf{h}, \boldsymbol{\theta})$. The parameters a_t and s_t^2 are chosen so that $\ln g(\ln y_t^2|\widehat{h}_t, \boldsymbol{\theta})$ and $\ln f(\ln y_t^2|\widehat{h}_t, \boldsymbol{\theta})$ have equal first and second derivatives, where \widehat{h}_t is the smoothed value of h_t provided by the Kalman filter applied to the approximating model. Remark that this is a different criterion from that used in EIS. Finally, $g(\mathbf{h}|\mathbf{y}, \boldsymbol{\theta})$ can be simulated with the Gaussian simulation smoother of de Jong and Shephard (1995).

In brief, the likelihood function is approximated by

$$g(\mathbf{y}|\boldsymbol{\theta}) \frac{1}{R} \sum_{r=1}^R \frac{f(\mathbf{y}|\mathbf{h}^r, \boldsymbol{\theta})}{g(\mathbf{y}|\mathbf{h}^r, \boldsymbol{\theta})}, \quad (2.32)$$

where $\mathbf{h}^r = \{h_t^r\}_{t=1}^T$ is simulated independently R times with the importance sampler and $g(\mathbf{y}|\boldsymbol{\theta})$ is computed by the Kalman filter. Equations (2.31) and (2.32) show that importance sampling serves to evaluate the departure of the actual likelihood from the likelihood of the approximating model. R is fixed to 250 in practice.

For SML estimation, the approximation in (2.32) is transformed in logarithm. This induces a bias since the expectation of the log of the sample mean is not the log of the corresponding integral in (2.31). The bias is corrected by adding $s_w^2/(2R\bar{w})$ to the log of (2.32), where s_w^2 is the sample variance of the ratios $w^r = f(\mathbf{y}|\mathbf{h}^r, \boldsymbol{\theta})/g(\mathbf{y}|\mathbf{h}^r, \boldsymbol{\theta})$ and \bar{w} is the sample mean of the same ratios, i.e. \bar{w} is the sample mean appearing in (2.32). Moreover, Durbin and Koopman (1997) use antithetic and control variables to improve the efficiency of the estimator of the log-likelihood function.

Durbin and Koopman (2000) present several generalizations of MCML (e.g. the case where the state variable in non-Gaussian) and develop analogous methods for Bayesian inference.

MCMC (Kim et al. (1998))

We present briefly the ‘Mixture Sampler’, one of the three algorithms added by Kim et al. (1998) to the six algorithms already in the literature at that time (see their paper for references). They approximate the density of $\varepsilon_t = \ln u_t^2$ by a finite mixture of seven Gaussian densities, such that in particular the first four moments of both densities are equal. The approximating density can be written as

$$f_a(\varepsilon_t) = \sum_{i=1}^7 \Pr[s_t = i]f(\varepsilon_t|s_t = i) = \sum_{i=1}^7 \Pr[s_t = i]f_N(\varepsilon_t|b_i - 1.2704, c_i^2) , \quad (2.33)$$

where s_t is a discrete random variable, while $\Pr[s_t = i]$, b_i and c_i are known constants (independent of t). The constant -1.2704 is the expected value of a $\ln \chi^2(1)$ variable.

The crux of the algorithm is to add $s = \{s_i\}_{i=1}^n$ to θ and h in the MCMC sampling space. This makes it possible to sample $h|s, \theta, y$, $s|h, y$ and $\theta|h, y$ within a Gibbs sampling algorithm. Remark that s and θ are independent given h and y . Moreover, h can be sampled entirely as a vector. The intuition behind this property is that, once s is known, the relevant term of the mixture (2.33) is known for each observation, and since this is a Gaussian density, the whole apparatus of the Kalman filter can be used. Actually, this a bit more involved since the relevant Gaussian density depends on t , but an augmented Kalman filter is available for this case.

Sampling h as one block is a big progress over previous algorithms, such as in Jacquier et al. (1994), where each element h_t is sampled individually given the other elements of h (plus θ and y). The slow convergence of such algorithms is due to the high correlations between the elements of h .

Kim et al. (1998) write the model in state space form, using μ rather than ω or τ as a parameter, i.e.

$$\ln y_t^2 = h_t + \varepsilon_t , \quad (2.34)$$

$$h_t - \mu = \beta(h_{t-1} - \mu) + \sigma v_t . \quad (2.35)$$

The ‘Mixture Sampler’ algorithm is summarized in Table 2.7. Notice that once θ has been sampled, it is easy to transform the draws of μ into equivalent draws of ω or τ by using the relationships between these parameters. Since inference is Bayesian, prior densities must be specified. For σ^2 , an inverted gamma prior density is convenient since the conditional posterior is in the same class and

Table 2.7. Summary of ‘Mixture Sampler’ algorithm

Parameter	Conditional posterior or sampling method
h	Gaussian simulation smoother
s	Univariate discrete distribution for each s_t
σ^2	Inverted gamma distribution
β	Rejection or Metropolis-Hastings sampler
μ	Normal distribution

easy to simulate. For β , any prior can be used since the conditional posterior is approximated and rejection sampling is used. A beta prior density is advocated by Kim et al. (1998). For μ , a Gaussian or uninformative prior results in a Gaussian conditional posterior.

Kim et al. (1998) also propose an algorithm to compute filtered estimates of h_t , from which model diagnostics can be obtained as described above for EIS.

Application

2.3.3

For illustration, estimates of the canonical SV model parameters are reported in Table 2.8 for a series of 6107 centred daily returns of the Standard and Poor's 500 (SP500) composite price index (period: 02/01/80–30/05/03, source: Datastream). Returns are computed as 100 times the log of the price ratios. The sample mean and standard deviation are equal to 0.03618 and 1.0603, respectively.

Table 2.8. ML and Bayesian estimates of SV model (2.19)

	EIS (ω)		MCML (τ)		MCMC (τ)	
ω/τ	-0.00524	(0.00227)	0.863	(0.0469)	0.864	(0.0494)
β	0.982	(0.00385)	0.982	(0.00389)	0.983	(0.00382)
σ	0.149	(0.0138)	0.147	(0.0135)	0.143	(0.0139)
llf	-8023.98		-8023.80			
Time	2.36 min		7.56 min		6.23 min	
Code	Gauss		Ox		Ox	

llf: value of log-likelihood function at the reported estimate;
EIS, MCML, and MCMC are defined in Sect. 2.3.2

We used computer codes provided by the authors cited above. For EIS, we received the code from R. Liesenfeld, for MCML and MCMC we downloaded them from the web site staff.feweb.vu.nl/koopman/sv.

For SML estimation by EIS or MCML, identical initial values ($\beta = 0.96$, $\sigma = 0.15$, $\omega = 0.02$ or $\tau = 0.01$) and optimization algorithms (BFGS) are used, but in different programming environments. Therefore, the computing times are not fully comparable, although a careful rule of thumb is that Ox is two to three times faster than Gauss (see Cribari-Neto (1997)). Reported execution times imply that EIS appears to be at least six times faster than MCML. This is a reversal of a result reported by Sandman and Koopman (1998, p. 289), but they compared MCML with a precursor of EIS implemented by Danielson (1994). More importantly, the two methods deliver quasi-identical results.

MCMC results are based on 18,000 draws after dropping 2000 initial draws. The posterior means and standard deviations are also quite close to the ML results. The posterior density of σ (computed by kernel estimation) is shown in Fig. 2.1 together with the large sample normal approximation to the density of the ML estimator using the EIS results. The execution time for MCMC is difficult to compare with

the other methods since it depends on the number of Monte Carlo draws. It is however quite competitive since reliable results are obtained in no more time than MCML in this example.

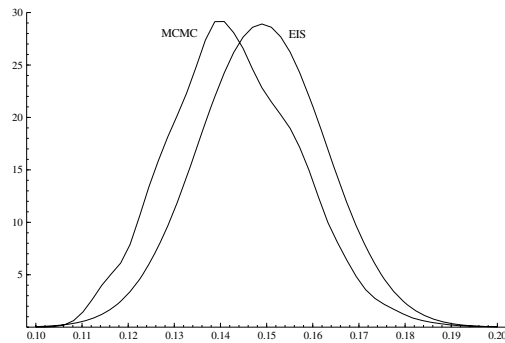


Figure 2.1. Posterior density of σ and normal density of the MLE

2.3.4 Extensions of the Canonical SV Model

The canonical model presented in (2.19) is too restrictive to fit the excess kurtosis of many return series. Typically, the residuals of the model reveal that the distribution of u_t has fatter tails than the Gaussian distribution. The assumption of normality is most often replaced by the assumption that $u_t \sim t(0, 1, \nu)$, which denotes Student- t distribution with zero mean, unit variance, and degrees of freedom parameter $\nu > 2$. SML estimates of ν are usually between 5 and 15 for stock and foreign currency returns using daily data. Posterior means are larger because the posterior density of ν has a long tail to the right.

Several other extensions of the simple SV model presented in (2.19) exist in the literature. The mean of y_t need not be zero and may be a function of explanatory variables x_t (often a lag of y_t and an intercept term). Similarly h_t may be a function of observable variables (z_t) in addition to its own lags. An extended model along these lines is

$$\begin{aligned} y_t &= x_t^T \gamma + \exp(h_t/2) u_t, \\ h_t &= \omega + z_t^T \alpha + \beta h_{t-1} + \sigma v_t. \end{aligned} \quad (2.36)$$

It should be obvious that all these extensions are very easy to incorporate in EIS (see Liesenfeld and Richard (2003)) and MCML (see Sandman and Koopman (1998)). Bayesian estimation by MCMC remains quite usable but becomes more demanding in research time to tailor the algorithm for achieving a good level of efficiency of the Markov chain (see Chib et al. (2002), in particular p 301–302, for such comments).

Chib et al. (2002) also include a jump component term $k_t q_t$ in the conditional mean part to allow for irregular, transient movements in returns. The random

variable q_t is equal to 1 with unknown probability κ and zero with probability $1 - \kappa$, whereas k_t is the size of the jump when it occurs. These time-varying jump sizes are assumed independent draws of $\ln(1 + k_t) \sim N(-0.5\delta^2, \delta^2)$, δ being an unknown parameter representing the standard deviation of the jump size. For daily SP500 returns (period: 03/07/1962–26/08/1997) and a Student- t density for u_t , Chib et al. (2002) report posterior means of 0.002 for κ , and 0.034 for δ (for prior means of 0.02 and 0.05, respectively). This implies that a jump occurs on average every 500 days, and that the variability of the jump size is on average 3.4%. They also find that the removal of the jump component from the model reduces the posterior mean of ν from 15 to 12, which corresponds to the fact that the jumps capture some outliers.

Another extension consists of relaxing the restriction of zero correlation (ρ) between u_t and v_t . This may be useful for stock returns for which a negative correlation corresponds to the leverage effect of the financial literature. If the correlation is negative, a drop of u_t , interpreted as a negative shock on the return, tends to increase v_t and therefore h_t . Hence volatility increases more after a negative shock than after a positive shock of the same absolute value, which is a well-known stylized fact. Sandman and Koopman (1998) estimate such a model by MCML, and report $\hat{\rho} = -0.38$ for daily returns of the SP500 index (period: 02/01/80–30/12/87), while Jacquier et al. (2004) do it by Bayesian inference using MCMC and report a posterior mean of ρ equal to -0.20 on the same data. They use the same reparametrization as in (2.13) to impose that the first diagonal element of the covariance matrix of u_t and σv_t must be equal to 1. This covariance matrix is given by

$$\Sigma = \begin{pmatrix} 1 & \rho\sigma \\ \rho\sigma & \sigma^2 \end{pmatrix} = \begin{pmatrix} 1 & \psi \\ \psi & \phi^2 + \psi^2 \end{pmatrix}, \quad (2.37)$$

where the last matrix is a reparametrization. This enables to use a normal prior on the covariance ψ and an inverted gamma prior on ϕ^2 , the conditional variance of σv_t given u_t . The corresponding conditional posteriors are of the same type, so that simulating these parameters in the MCMC algorithm is easy. This approach can also be used if u_t has a Student- t distribution.

Multivariate SV models are also on the agenda of researchers. Liesenfeld and Richard (2003) estimate by EIS a one-factor model introduced by Shephard (1996), using return series of four currencies. Kim et al. (1998), Sect. 6.6, explain how to deal with the multi-factor model case by extending the MCMC algorithm reviewed in Sect. 2.3.2.

Stochastic Duration and Intensity Models

Models akin to the SV model have been used for dynamic duration analysis by Bauwens and Veredas (2004) and Bauwens and Hautsch (2003). The context of application is the analysis of a sequence of time spells between events (also called

durations) occurring on stock trading systems like the New York Stock Exchange (NYSE). Time stamps of trades are recorded for each stock on the market during trading hours every day, resulting in an ordered series of durations. Marks, such as the price, the exchanged quantity, the prevailing bid and ask prices, and other observed features may also be available, enabling to relate the durations to the marks in a statistical model. See Bauwens and Giot (2001) for a presentation of the issues.

Let $0 = t_0 < t_1 < t_2 < \dots < t_n$ denote the arrival times and $d_1, d_2 \dots d_n$ denote the corresponding durations, i.e. $d_i = t_i - t_{i-1}$. The stochastic conditional duration (SCD) model of Bauwens and Veredas (2004) is defined as

$$\begin{aligned} d_i &= \exp(\psi_i) u_i, \quad u_i \sim D(\gamma), \quad t = 1, \dots, n, \\ \psi_i &= \omega + \beta \psi_{i-1} + \sigma v_i, \quad v_i \sim N(0, 1), \end{aligned} \quad (2.38)$$

where $D(\gamma)$ denotes some distribution on the positive real line, possibly depending on a parameter γ . For example, Bauwens and Veredas use the Weibull distribution and the gamma distribution (both with shape parameter denoted by γ). Assuming that the distribution of u_i is parameterized so that $E(u_i) = 1$, ψ_i is the logarithm of the unobserved mean of d_i , and is modelled by a Gaussian autoregressive process of order one. It is also assumed that $\{u_i\}$ and $\{v_i\}$ are mutually independent sequences. The parameters to be estimated are $(\omega, \beta, \sigma, \gamma)$, jointly denoted θ . The parameter space is $\mathbb{R} \times (-1, 1) \times \mathbb{R}_+ \times \mathbb{R}_+$.

The similarity with the canonical SV model (2.19) is striking. A major difference is the non-normality of u_i since this is by definition a positive random variable. This feature makes it possible to identify γ . Therefore, the estimation methods available for the SV model can be adapted to the estimation of SCD models. Bauwens and Veredas (2004) have estimated the SCD model by the quasi-maximum likelihood (QML) method, since the first equation of the model may be expressed as $\ln d_i = \psi_i + \ln u_i$. If $\ln u_i$ were Gaussian, the model would be a Gaussian linear state space model and the Kalman filter could be directly applied. QML relies on maximizing the likelihood function as if $\ln u_i$ were Gaussian. The QML estimator is known to be consistent but inefficient relative to the ML estimator which would obtain if the correct distribution of $\ln u_i$ were used to define the likelihood function. Galli (2003), Chap. 3, has studied by simulation the loss of efficiency of QML relative to ML. ML estimation assuming a Weibull distribution is done by applying the EIS algorithm. For a sample size of 500 observations, the efficiency loss ranges from 20 to 50%, except for the parameter ω , for which it is very small. He also applied the EIS method using the same data as in Bauwens and Veredas (2004). For example, for a dataset of 1576 volume durations of the Boeing stock (period: September–November 1996; source: TAQ database of NYSE), the ML estimates are: $\widehat{\omega} = -0.028$, $\widehat{\beta} = 0.94$, $\widehat{\sigma}^2 = 0.0159$, $\widehat{\gamma} = 1.73$. They imply a high persistence in the conditional mean process (corresponding to duration clustering), a Weibull distribution with an increasing concave hazard function, and substantial heterogeneity. Notice that an interesting feature of the

SCD model is that the distribution of u_i conditional to the past information, but marginalized with respect to the latent process, is a Weibull mixed by a lognormal distribution.

Strickland et al. (2003) have designed a MCMC algorithm for the SCD model (2.38) assuming a standard exponential distribution for u_i . The design of their MCMC algorithm borrows features from Koopman and Durbin's MCML approach and one of the MCMC algorithms used for the SV model.

As an alternative to modelling the sequence of durations, Bauwens and Hautsch (2003) model directly the arrival times through the intensity function of the point process. Their model specifies a dynamic intensity function, where the intensity function is the product of two intensity components: an observable component that depends on past arrival times, and a latent component. The logarithm of the latter is a Gaussian autoregressive process similar to the second equation in (2.19) and (2.38). The observable component may be a Hawkes process (Hawkes (1971)) or an autoregressive intensity model (Russell (1999)). When the model is multivariate, there is an observable intensity component specific to each particular point process, while the latent component is common to all particular processes. Interactions between the processes occur through the observable components and through the common component. The latter induces similar dynamics in the particular processes, reflecting the impact of a common factor influencing all processes. Bauwens and Hautsch use intensity-based likelihood inference, with the EIS algorithm to deal with the latent component.

Finite Mixture Models

Many econometric issues require models that are richer or more flexible than the conventional regression type models. Several possibilities exist. For example, as explained in Sect. 2.2.3, the logit model is made more realistic by generalizing it to a mixed logit. Many models currently used in econometrics can be generalized in such a way.

In this section, we assume that the univariate or multivariate observations y_j are considered as draws of

$$\tilde{f}(y_j) = \sum_{g=1}^G \eta_g f(y_j | \theta_g) \quad (2.39)$$

with $\eta_1 + \dots + \eta_G = 1$. The densities $f(\cdot | \theta_g)$ are called component distributions. The observation y_j comes from one of these component distributions but we do not observe to which component it belongs. The mixture problem involves making inference about the η_g 's and the parameters of the component distributions given only a sample from the mixture. The closer the component distributions are to each other, the more difficult this is because of problems of identifiability and computational instability.

2.4.1 Inference and Identification

The structure of (2.39) implies that the likelihood for all the J observations contains G^J terms

$$L(\boldsymbol{\eta}, \boldsymbol{\theta} | \mathbf{y}) \propto \prod_{j=1}^J \left(\sum_{g=1}^G \eta_{gf} (y_j | \boldsymbol{\theta}_g) \right), \quad (2.40)$$

where $\boldsymbol{\eta} = (\eta_1, \dots, \eta_G)^T$ and $\boldsymbol{\theta} = (\boldsymbol{\theta}_1, \dots, \boldsymbol{\theta}_G)^T$ contain all the parameters and \mathbf{y} denotes all the data. Maximum likelihood estimation using numerical optimization techniques, requiring many evaluations of the likelihood function, becomes cumbersome, if not unfeasible, for large G and J . This is even worse for multivariate observations.

Bayesian inference on finite mixture distributions by MCMC sampling is explained in Diebolt and Robert (1994). Gibbs sampling on $(\boldsymbol{\eta}, \boldsymbol{\theta})$ is difficult since the posterior distributions of $\boldsymbol{\eta} | \boldsymbol{\theta}, \mathbf{y}$ and $\boldsymbol{\theta} | \boldsymbol{\eta}, \mathbf{y}$ are generally unknown. For the same reason as for the probit model in Sect. 2.2.1 and the stochastic volatility model in Sect. 2.3, inference on the finite mixture model is straightforward once the state or group of an observation is known. Data augmentation is therefore an appropriate way to render inference easier. Define the state indicator S_j which takes value $s_j = g$ when y_j belongs to state or group g where $g \in \{1, \dots, G\}$. Denote by \mathbf{S} the J -dimensional discrete vector containing all the state indicators. To facilitate the inference, prior independence, that is $\varphi(\boldsymbol{\eta}, \boldsymbol{\theta}, \mathbf{S}) = \varphi(\boldsymbol{\eta})\varphi(\boldsymbol{\theta})\varphi(\mathbf{S})$, is usually imposed. As shown in the next examples, the posterior distributions $\mathbf{S} | \boldsymbol{\eta}, \boldsymbol{\theta}, \mathbf{y}$, $\boldsymbol{\theta} | \boldsymbol{\eta}, \mathbf{S}, \mathbf{y}$ and $\boldsymbol{\eta} | \boldsymbol{\theta}, \mathbf{S}, \mathbf{y}$ are either known distributions easy to sample from or they are distributions for which a second, but simpler, MCMC sampler is set up. A Gibbs sampler with three main blocks may therefore be used.

The complete data likelihood of the finite mixture is invariant to a relabeling of the states. This means that we can take the labeling $\{1, 2, \dots, G\}$ and do a permutation $\{\varphi(1), \varphi(2), \dots, \varphi(G)\}$ without changing the value of the likelihood function. If the prior is also invariant to relabeling then the posterior has this property also. As a result, the posterior has potentially $G!$ different modes. To solve this identification or label switching problem, identification restrictions have to be imposed.

Note that the inference described here is conditional on G , the number of components. There are two modelling approaches to take care of G . First, one can treat G as an extra parameter in the model as is done in Richardson and Green (1997) who make use of the reversible jump MCMC methods. In this way, the prior information on the number of components can be taken explicitly into account by specifying for example a Poisson distribution on G in such a way that it favors a small number of components. A second approach is to treat the choice of G as a problem of model selection. By so-doing one separates the issue of the choice of G from estimation with G fixed. For example, one can take $G = 2$ and $G = 3$ and do the estimation separately for the two models. Then Bayesian model comparison techniques (see Chap. III.11) can be applied, for instance by the calculation of the Bayes factor, see Cowles and Carlin (1996) and Chib (1995) for more details.

Examples

2.4.2

We review two examples. The first example fits US quarterly GNP data using a Markov switching autoregressive model. The second example is about the clustering of many GARCH models.

Markov Switching Autoregressive Model

Frühwirth-Schnatter (2001) uses US quarterly real GNP growth data from 1951:2 to 1984:4. This series was initially used by Hamilton (1989) and is displayed in Fig. 2.2. The argument is that contracting and expanding periods are generated by the same model but with different parameters. These models are called state- (or regime-) switching models.

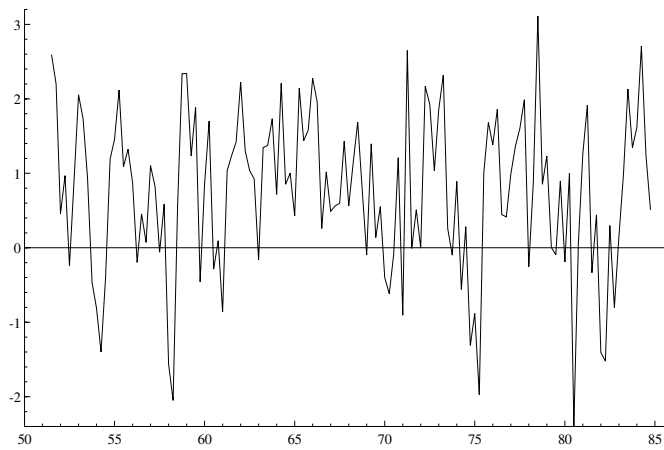


Figure 2.2. US real GNP growth data in percentages (1951:2 to 1984:4)

After some investigation using Bayesian model selection techniques, the adequate specification for the US growth data is found to be the two-state switching AR(2) model

$$y_t = \beta_{i,1}y_{t-1} + \beta_{i,2}y_{t-2} + \beta_{i,3} + \varepsilon_{t,i} \quad \varepsilon_{t,i} \sim N(0, \sigma_i^2) \quad i = 1, 2. \quad (2.41)$$

The idea behind the choice of two states is motivated by the contracting (negative growth) and expanding periods (positive growth) in an economy. The conditional posteriors for the σ_i^2 's are independent inverted gamma distributions. For the β_i 's, the conditional posteriors are independent normal distributions. Inference for the switching model in (2.41) is done in two steps. The first step is to construct an MCMC sample by running the random permutation sampler. Generally speaking, a draw from the random permutation sampler is obtained as follows:

- (1) Draw from the model by use of the Gibbs sampler for example.
- (2) Relabel the states randomly.

By so-doing, one samples from the unconstrained parameter space with balanced label switching. Note that in (2), there are $G!$ possibilities to relabel when there are G possible states.

In the second step, this sample is used to identify the model. This is done by visual inspection of the posterior marginal and bivariate densities. Identification restrictions need to be imposed to avoid multimodality of the posterior densities. Once suitable restrictions are found, a final MCMC sample is constructed to obtain the moments of the constrained posterior density. The latter sample is constructed by permutation sampling under the restrictions, which means that (2) is replaced by one permutation defining the constrained parameter space.

In the GNP growth data example, two identification restrictions seem possible, namely $\beta_{1,1} < \beta_{2,1}$ and $\beta_{1,3} < \beta_{2,3}$, see Frühwirth-Schnatter (2001) for details. Table 2.9 provides the posterior means and standard deviations of the β_{ij} 's for both identification restrictions.

Table 2.9. Posterior means and standard deviations

	$\beta_{1,1} < \beta_{2,1}$		$\beta_{1,3} < \beta_{2,3}$	
	Contraction	Expansion	Contraction	Expansion
$\beta_{i,1}$	0.166 (0.125)	0.33 (0.101)	0.249 (0.164)	0.295 (0.116)
$\beta_{i,2}$	0.469 (0.156)	-0.129 (0.093)	0.462 (0.164)	-0.114 (0.098)
$\beta_{i,3}$	-0.479 (0.299)	1.07 (0.163)	-0.557 (0.322)	1.06 (0.175)

The GNP growth in contraction and expansion periods not only have different unconditional means, they are also driven by different dynamics. Both identification restrictions result in similar posterior moments.

Clustering of Many GARCH Models

Bauwens and Rombouts (2003) focus on the differentiation between the component distributions via different conditional heteroskedasticity structures by the use of GARCH models. In this framework, the observation y_j is multivariate and the θ_g 's are the parameters of GARCH(1,1) models. The purpose is to estimate many, of the order of several hundreds, GARCH models. Each financial time series belongs to one of the G groups but it is not known a priori which series belongs to which cluster.

An additional identification problem arises due to the possibility of empty groups. If a group is empty then the posterior of θ_g is equal to the prior of θ_g . Therefore an improper prior is not allowed for θ_g . The identification problems are solved by using an informative prior on each θ_g . The identification restrictions use the fact that we work with GARCH models: we select rather non-overlapping supports for the parameters, such that the prior $\varphi(\theta) = \prod_{g=1}^G \varphi(\theta_g)$ depends on a labeling choice. Uniform prior densities on each parameter, on finite intervals, possibly subject to stationarity restrictions, are relatively easy to specify.

Bayesian inference is done by use of the Gibbs sampler and data augmentation. Table 2.10 summarizes the three blocks of the sampler.

Table 2.10. Summary of conditional posteriors

Parameter	Conditional posterior or sampling method
S	Multinomial distribution
η	Dirichlet distribution
θ	Griddy-Gibbs sampler

Because of the prior independence of the θ_g 's, the griddy-Gibbs sampler is applied separately G times.

As an illustration we show the posterior marginals of the following model

$$\tilde{f}(y_j) = \sum_{g=1}^3 \eta_g f(y_j | \theta_g) \quad (2.42)$$

with $\eta_1 = 0.25$, $\eta_2 = 0.5$, $J = 100$ and $T_j = 1000$. The components are defined more precisely as

$$f(y_j | \theta_g) = \prod_{t=1}^{T_j} f(y_{j,t} | \theta_g, I_{j,t}) \quad (2.43)$$

$$y_{j,t} | \theta_g, I_{j,t} \sim N(0, h_{j,t}) \quad (2.44)$$

$$h_{j,t} = (1 - \alpha_g - \beta_g) \tilde{\omega}_j + \alpha_g (y_{j,t-1})^2 + \beta_g h_{j,t-1}, \quad (2.45)$$

where $I_{j,t}$ is the information set until $t - 1$ containing (at least) $y_{j,1}, \dots, y_{j,t-1}$ and initial conditions which are assumed to be known. For the simulation of the data $\tilde{\omega}_j$ is fixed equal to one which implies that the unconditional variance for every generated data series is equal to one. However, the constant $\tilde{\omega}_j$ in the conditional variance is not subject to inference, rather it is fixed at the empirical variance of the data. Table 2.11 presents the true values, the prior intervals on the θ_g 's and posterior results on η and θ .

Bauwens and Rombouts (2003) successfully apply this model to return series of 131 US stocks. Comparing the marginal likelihoods for different models, they find that $G = 3$ is the appropriate choice for the number of component distributions.

Other interesting examples of finite mixture modelling exist in the literature. Frühwirth-Schnatter and Kaufmann (2002) develop a regime switching panel data model. Their purpose is to cluster many short time series to capture asymmetric effects of monetary policy on bank lending. Deb and Trivedi (1997) develop a finite mixture negative binomial count model to estimate six measures of medical care demand by the elderly. Chib and Hamilton (2000) offer a flexible Bayesian

Table 2.11. Posterior results on η and θ ($G = 3$)

		η_1	η_2	η_3
True value		0.25	0.50	0.25
Mean		0.2166	0.4981	0.2853
Standard deviation		0.0555	0.0763	0.0692
Correlation matrix		1	-0.4851	-0.2677
		-0.4851	1	-0.7127
		-0.2677	-0.7127	1

		$g = 1$	$g = 2$	$g = 3$
True value	α_g	0.04	0.12	0.20
	β_g	0.90	0.60	0.40
Prior interval	α_g	0.001,0.07	0.07,0.15	0.15,0.25
	β_g	0.65,0.97	0.45,0.75	0.20,0.60
Mean	α_g	0.0435	0.1041	0.1975
	β_g	0.8758	0.5917	0.4369
Standard deviation	α_g	0.0060	0.0092	0.0132
	β_g	0.0238	0.0306	0.0350
Correlation	α_g, β_g	-0.7849	-0.71409	-0.7184

analysis of the problem of causal inference in models with non-randomly assigned treatments. Their approach is illustrated using hospice data and hip fracture data.

References

- Allenby, G. and Rossi, P. (1999). Marketing models of consumer heterogeneity. *Journal of Econometrics*, 89:57–78.
- Bauwens, L. and Giot, P. (2001). *Econometric Modelling of Stock Market Intraday Activity*. Kluwer.
- Bauwens, L. and Hautsch, N. (2003). Dynamic latent factor models for intensity processes. CORE DP 2003/103.
- Bauwens, L. and Rombouts, J. (2003). Bayesian clustering of many GARCH models. CORE DP 2003/87.
- Bauwens, L. and Veredas, D. (2004). The stochastic conditional duration model: a latent factor model for the analysis of financial durations. Forthcoming in *Journal of Econometrics*.
- Bhat, C. (2001). Quasi-random maximum simulated likelihood estimation of the mixed multinomial logit model. *Transportation Research Part B*, 35:677–693.
- Bollerslev, T., Engle, R., and Nelson, D. (1994). ARCH models. In Engle, R. and McFadden, D., editors, *Handbook of Econometrics*, Chap. 4, pages 2959–3038. North Holland Press, Amsterdam.

- Brownstone, D. and Train, K. (1999). Forecasting new product penetration with flexible substitution patterns. *Journal of Econometrics*, 89:109–129.
- Chib, S. (1995). Marginal likelihood from the Gibbs output. *Journal of the American Statistical Association*, 90:1313–1321.
- Chib, S. and Greenberg, E. (1998). Analysis of multivariate probit models. *Biometrika*, 85:347–361.
- Chib, S. and Hamilton, B. (2000). Bayesian analysis of cross-section and clustered data treatment models. *Journal of Econometrics*, 97:25–50.
- Chib, S., Nardari, F., and Shephard, N. (2002). Markov chain Monte Carlo methods for stochastic volatility models. *Journal of Econometrics*, 108:291–316.
- Chintagunta, P. and Honore, B. (1996). Investigating the effects of marketing variables and unobserved heterogeneity in a multinomial probit model. *International Journal of Research in Marketing*, 13:1–15.
- Cowles, M. and Carlin, B. (1996). Markov chain Monte Carlo convergence diagnostics: A comparative review. *Journal of the American Statistical Association*, 91:883–904.
- Cribari-Neto, F. (1997). Econometric programming environments: Gauss, Ox and S-plus. *Journal of Applied Econometrics*, 12:77–89.
- Danielson, J. (1994). Stochastic volatility in asset prices: estimation with simulated maximum likelihood. *Journal of Econometrics*, 61:375–400.
- de Jong, P. and Shephard, N. (1995). The simulation smoother for time series models. *Biometrika*, 82:339–350.
- Deb, P. and Trivedi, P. (1997). Demand for medical care by the elderly: A finite mixture approach. *Journal of Applied Econometrics*, 12:313–336.
- Diebolt, J. and Robert, C. (1994). Estimation of finite mixture distributions through Bayesian sampling. *Journal of the Royal Statistical Society, Series B*, 56:363–375.
- Durbin, J. and Koopman, S. (1997). Monte Carlo maximum likelihood estimation for non-Gaussian state space models. *Biometrika*, 84:669–684.
- Durbin, J. and Koopman, S. (2000). Time series analysis of non-Gaussian observations based on state space models from both classical and Bayesian perspectives. *Journal of the Royal Statistical Society B*, 62:3–56.
- Franses, P. and Paap, R. (2001). *Quantitative Models in Marketing Research*. Cambridge University Press, Cambridge.
- Frühwirth-Schnatter, S. (2001). Markov chain Monte Carlo estimation of classical and dynamic switching and mixture models. *Journal of the American Statistical Association*, 96:194–209.
- Frühwirth-Schnatter, S. and Kaufmann, S. (2002). Bayesian clustering of many short time series. Working Paper, Vienna University of Economics and Business Administration.
- Galli, F. (2003). *Econometria dei Dati Finanziari ad Alta Frequenza*. Tesi di Dottorato in Economia Politica, Bologna.
- Geweke, J., Keane, M., and Runkle, D. (1997). Statistical inference in the multinomial multiperiod probit model. *Journal of Econometrics*, 80:125–165.

- Ghysels, E., Harvey, A., and Renault, E. (1996). Stochastic volatility. In Maddala, G. and Rao, C., editors, *Handbook of Statistics*, pages 119–191. Elsevier Science, Amsterdam.
- Gourieroux, C. and Monfort, A. (1997). *Simulation-based Econometric Methods*. Oxford University Press, Oxford.
- Hajivassiliou, V. and Mc Fadden, D. (1998). The method of simulated scores for the estimation of LDV models. *Econometrica*, 66:863–896.
- Hajivassiliou, V. and Ruud, P. (1994). *Classical estimation methods for LDV models using simulation*. In: *Handbook of Econometrics*, Vol. 4, Chap. 40, North Holland, Amsterdam.
- Hamilton, J. (1989). A new approach to the economic analysis of nonstationary time series and the business cycle. *Econometrica*, 57:357–384.
- Hawkes, A. G. (1971). Spectra of some self-exciting and mutually exciting point processes. *Biometrika*, 58:83–90.
- Horowitz, J. (1998). *Semiparametric Methods in Econometrics*. Springer Verlag, Berlin.
- Jacquier, E., Polson, N., and Rossi, P. (1994). Bayesian analysis of stochastic volatility models (with discussion). *Journal of Business and Economic Statistics*, 12:371–417.
- Jacquier, E., Polson, N., and Rossi, P. (2004). Bayesian analysis of stochastic volatility models with fat-tails and correlated errors. Forthcoming in *Journal of Econometrics*.
- Kim, S., Shephard, N., and Chib, S. (1998). Stochastic volatility: likelihood inference and comparison with ARCH models. *Review of Economic Studies*, 65:361–393.
- Liesenfeld, R. and Richard, J.-F. (2003). Univariate and multivariate stochastic volatility models: estimation and diagnostics. *Journal of Empirical Finance*, 10:505–531.
- Maddala, G. (1983). *Limited-dependent and Qualitative Variables in Econometrics*. Cambridge University Press, Cambridge.
- Mariano, R., Schuermann, T., and Weeks, M. (2000). *Simulation-based Inference in Econometrics*. Cambridge University Press, Cambridge.
- McCulloch, R., Polson, N., and Rossi, P. (2000). A Bayesian analysis of the multinomial probit model with fully identified parameters. *Journal of Econometrics*, 99:173–193.
- McFadden, D. and Train, K. (2000). Mixed MNL models for discrete response. *Journal of Applied Econometrics*, 15:447–470.
- Paap, R. and Franses, P. (2000). A dynamic multinomial probit model for brand choice with different long-run and short-run effects of marketing-mix variables. *Journal of Applied Econometrics*, 15:717–744.
- Pagan, A. and Ullah, A. (1999). *Nonparametric Econometrics*. Cambridge University Press, Cambridge.
- Richard, J.-F. (1998). Efficient high-dimensional Monte Carlo importance sampling. Manuscript, University of Pittsburgh.

-
- Richardson, S. and Green, P. (1997). On Bayesian analysis of mixtures with an unknown number of components. *Journal of the Royal Statistical Society, Series B*, 59:731–792.
- Russell, J. (1999). Econometric modeling of multivariate irregularly-spaced high-frequency data. Manuscript, University of Chicago.
- Sandman, G. and Koopman, S. (1998). Estimation of stochastic volatility models via Monte Carlo maximum likelihood. *Journal of Econometrics*, 67:271–301.
- Shephard, N. (1996). *Statistical aspects of ARCH and stochastic volatility*, Chap. 1, pages 1–67. Time Series Models: In Econometrics, Finance and Other Fields. Chapman & Hall, in D. R. Cox, D. V. Hinkley, and O. E. Barndorff-Nielsen (eds.), London.
- Strickland, C., Forbes, C., and Martin, G. (2003). Bayesian analysis of the stochastic conditional duration model. Manuscript, Monash University.
- Tanner, M. and Wong, W. (1987). The calculation of posterior distributions by data augmentation. *Journal of the American Statistical Association*, 82:528–540.
- Train, K. (2003). *Discrete Choice Methods with Simulation*. Cambridge University Press, Cambridge.

Statistical and Computational Geometry of Biomolecular Structure

IV.3

Iosif I. Vaisman

3.1	<i>Introduction</i>	982
3.2	<i>Statistical Geometry of Molecular Systems</i>	983
3.3	<i>Tetrahedrality of Delaunay Simplices as a Structural Descriptor in Water</i>	986
3.4	<i>Spatial and Compositional Three-dimensional Patterns in Proteins</i>	987
3.5	<i>Protein Structure Comparison and Classification</i>	993
3.6	<i>Conclusions</i>	996

3.1 **Introduction**

Recent revolutionary developments in genomics and computational structural biology lead to the rapidly increasing amount of data on biomolecular sequences and structures. The deposition rate for both sequence and structure databases continues to grow exponentially. The efficient utilization of this data depends on the availability of methods and tools for biomolecular data analysis. Significant achievements have been made in DNA and protein sequence analysis, now the focus in bioinformatics research is shifting from sequence to structural and functional data. Accurate prediction of protein three-dimensional structure from its primary sequence represents one of the greatest challenges of modern theoretical biology. Detailed knowledge of protein structure is essential for understanding the mechanisms of biological processes at molecular, cellular, and evolutionary levels. The structures of only a fraction of all known primary sequences have been determined experimentally. Several approaches to protein structure prediction have been developed in recent years. Many of these approaches rely on the knowledge derived from the analysis of significant spatial and compositional patterns in known protein structures and understanding of the role these patterns play in the extremely complex processes, like protein folding or protein function. Such an analysis requires an objective definition of nearest neighbor residues that can be provided by the statistical geometry methods.

In the statistical geometry methods the nearest neighbor atoms or groups of atoms are identified by statistical analysis of irregular polyhedra obtained as a result of a specific tessellation in three-dimensional space. Voronoi tessellation partitions the space into convex polytopes called Voronoi polyhedra (Voronoi, 1908). For a molecular system the Voronoi polyhedron is the region of space around an atom, such that each point of this region is closer to the atom than to any other atom of the system. A group of four atoms whose Voronoi polyhedra meet at a common vertex forms another basic topological object called a Delaunay simplex (Delaunay, 1934). The results of the procedure for constructing Voronoi polyhedra and Delaunay simplices in two dimensions are illustrated in Fig. 3.1. The topological difference between these objects is that the Voronoi polyhedron represents the environment of individual atoms whereas the Delaunay simplex represents the ensemble of neighboring atoms. The Voronoi polyhedra and the Delaunay simplices are topological duals and they are completely determined by each other. However the Voronoi polyhedra may have different numbers of faces and edges, while the Delaunay simplices are always tetrahedra in three-dimensional space. These tetrahedra can be used to define objectively the nearest neighbor entities in molecular systems.

Delaunay tessellation is a canonical tessellation of space based on nearest neighbors (Aurenhammer, 1991; Sugihara, 1995). A Delaunay tessellation of a set of points is equivalent to a convex hull of the set in one higher dimension, it can be performed using the Quickhull algorithm developed by Barber et al. (1996). The Quickhull algorithm is a variation of the randomized, incremental algorithm of Clarkson

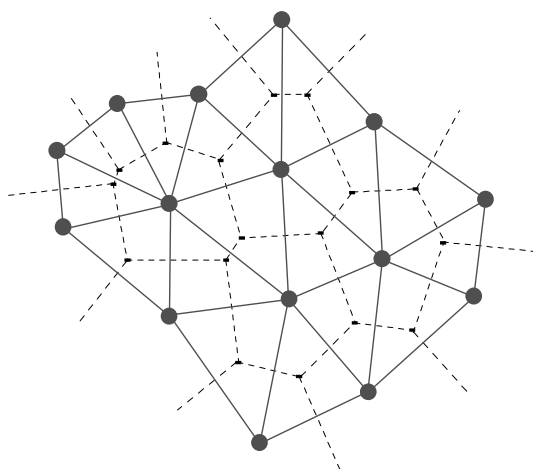


Figure 3.1. Voronoi/Delaunay tessellation in 2D space; Voronoi tessellation – *dashed line*, Delaunay tessellation – *solid line*

and Shor. The algorithm produces the Delaunay tessellation by computing the convex-hull of this set of points in four dimensions and is shown to be space and time efficient.

Statistical Geometry of Molecular Systems

3.2

A statistical geometry approach to study structure of molecular systems was pioneered by John Bernal, who in the late 1950s suggested that “many of the properties of liquids can be most readily appreciated in terms of the packing of irregular polyhedra” (Bernal, 1959). Bernal pointed out that “it would be most desirable to find the true minimum number of parameters or parametral functions defining the statistical structure of any homogenous irregular assemblage in the way that the lattice vectors define a regular one” (Bernal, 1959). Methods of computational geometry, Voronoi and Delaunay tessellations in particular, may be used to address this problem. This approach, based on the Voronoi partitioning of space (Voronoi, 1908) occupied by the molecule, was further developed by Finney for liquid and glass studies (Finney, 1970). Finney proposed a set of “descriptive parameters” for packing of polyhedra in simple liquids. In the mid-1970s the statistical geometry approach was first applied to study packing and volume distributions in proteins by Richards (1974), Chothia (1975) and Finney (1975).

Richards applied Voronoi tessellation to calculate atomic volumes in the experimentally solved crystallographic structures of ribonuclease C and lysozyme (Richards, 1974) and Chothia extended the calculations to a larger set of proteins

(Chothia, 1975). Standard Voronoi tessellation algorithm treats all atoms as points and allocates volume to each atom regardless of the atom size, which leads to the volume overestimate for the small atoms and underestimate for the large ones. Richards introduced changes in the algorithm (Richards, 1974) that made Voronoi volumes proportional to the atom sizes, creating chemically more relevant partitioning, however it has been done at the expense of the robustness of the algorithm. The Voronoi polyhedra in this case do not fill all available space. In addition to polyhedra around the atoms Richards' method produces so called vertex polyhedra in the unallocated volumes in the neighborhood of each vertex. As a result the accuracy of the tessellation is reduced. An alternative procedure, the "radical plane" partitioning, which is both chemically appropriate and completely rigorous was designed by Gellatly and Finney (1982) and applied to the calculation of protein volumes. The radical plane of two spheres is the locus of points from which the tangent lengths to both spheres are equal. Using the three-dimensional structure of RNAase-C as an example, they have shown that the differences between the results from the radical plane and Richards' methods are generally smaller than the difference between either of those and Voronoi's method. Both radical plane and Richards' methods are relatively insensitive to the treatment of surface, which makes them preferential to other techniques (Gellatly and Finney, 1982). Volume calculation remains one of the most popular applications of Voronoi tessellation to protein structure analysis. It has been used to evaluate the differences in amino acid residue volumes in solution and in the interior of folded proteins (Harpaz et al., 1994), to monitor the atomic volumes in the course of molecular dynamics simulation of a protein (Gerstein et al., 1995), to compare the sizes of atomic groups in proteins and organic compounds (Tsai et al., 1999), to calculate the atomic volumes on protein-protein interfaces (Lo Conte et al., 1999), and to measure sensitivity of residue volumes to the selection of tessellation parameters and protein training set (Tsai and Gerstein, 2002). Deviations from standard atomic volumes in proteins determined through Voronoi tessellation correlate with crystallographic resolution and can be used as a quality measure for crystal structures (Pontius et al., 1996). A modified Voronoi algorithm, where dividing planes between the atoms were replaced by curved surfaces, defined as the set of geometrical loci with equal orthogonal distance to the surfaces of the respective van der Waals spheres, was proposed by Goede et al. (1997). Voronoi cells with hyperbolic surface constructed by this algorithm improve the accuracy of volume and density calculations in proteins (Rother et al., 2003). Another extension of the Voronoi algorithm, the Laguerre polyhedral decomposition was applied to the analysis of the residue packing geometry (Sadoc et al., 2003).

One of the problems in constructing Voronoi diagram for the molecular systems is related to the difficulty of defining a closed polyhedron around the surface atoms, which leads to ambiguities in determining their volumes and densities (a recent example in Quillin and Matthews, 2000). This problem can be addressed by the "solvation" of the tessellated molecule in the at least one layer of solvent or by using computed solvent-accessible surface for the external closures of Voronoi polyhedra. The analysis of atomic volumes on the protein surface can be used to adjust param-

eters of the force field for implicit solvent models, where the solvent is represented by the generalized Born model of electrostatic solvation which require knowledge of the volume of individual solute atoms (Schaefer et al., 2001). Interactions between residues in proteins can be measured using the contact area between atoms defined as the area of intersection of the van der Waals sphere and the Voronoi polyhedron of an atom (Wernisch et al., 1999). Examining the packing of residues in proteins by Voronoi tessellation revealed a strong fivefold local symmetry similar to random packing of hard spheres, suggesting a condensed matter character of folded proteins (Soyer et al., 2000). Correlations between the geometrical parameters of Voronoi cells around residues and residue conformations were discovered by Angelov et al. (2002). Another recent study described application of Voronoi procedure to study atom-atom contacts in proteins (McConkey et al., 2002).

A topological dual to Voronoi partitioning, the Delaunay tessellation (Delaunay, 1934) has an additional utility as a method for non-arbitrary identification of neighboring points in the molecular systems represented by the extended sets of points in space. Originally the Delaunay tessellation has been applied to study model (Voloshin et al., 1989) and simple (Medvedev and Naberukhin, 1987) liquids, as well as water (Vaisman et al., 1993) and aqueous solutions (Vaisman and Berkowitz, 1992; Vaisman et al., 1994). The Delaunay tessellation proved to be a particularly convenient and efficient descriptor of water structure, where a natural tetrahedral arrangement of molecules is present in the first hydration shell (Vaisman et al., 1993).

The first application of the Delaunay tessellation for identification of nearest neighbor residues in proteins and derivation of a four-body statistical potential was developed in the mid-1990 s (Singh et al., 1996). This potential has been successfully tested for inverse protein folding (Tropsha et al., 1996), fold recognition (Zheng et al., 1997), decoy structure discrimination (Munson et al., 1997; Krishnamoorthy and Tropsha, 2003), protein design (Weberndorfer et al., 1999), protein folding on a lattice (Gan et al., 2001), mutant stability studies (Carter et al., 2001), computational mutagenesis (Masso and Vaisman, 2003), protein structure similarity comparison (Bostick and Vaisman, 2003), and protein structure classification (Bostick et al., 2004). Statistical compositional analysis of Delaunay simplices revealed highly nonrandom clustering of amino acid residues in folded proteins when all residues were treated separately as well as when they were grouped according to their chemical, structural, or genetic properties (Vaisman et al., 1998). A Delaunay tessellation based alpha-shape procedure for protein structure analysis was developed by Liang et al. (Liang et al., 1998a). Alpha shapes are polytopes that represent generalizations of the convex hull. A real parameter alpha defines the "resolution" of the shape of a point set (Edelsbrunner et al., 1983). Alpha shapes proved to be useful for the detection of cavities and pockets in protein structures (Liang et al., 1998b; 1998c). Several alternative Delaunay and Voronoi based techniques for cavity identification were described by Richards (1985), Bakowies and van Gunsteren (2002) and Chakravarty et al. (2002). Delaunay tessellation has been also applied to compare similarity of protein local substructures (Kobayashi and Go, 1997) and to study the mechanical response of a protein under applied pressure (Kobayashi et al., 1997).

Tetrahedrality of Delaunay Simplices as a Structural Descriptor in Water

Quantitative measurement of local structural order in the computational models of liquid water (and other associated liquids) is an intrinsically difficult problem. Conventional structure descriptors, such as radial distribution functions cannot be used to adequately evaluate structure in the specific regions of complex model systems like multicomponent solutions or solutions of large biological molecules (Vaisman and Berkowitz, 1992). Another set of structural descriptors, the geometric and thermodynamic parameters of hydrogen bond network depend on arbitrary values in the hydrogen bond definition (Vaisman et al., 1994). Statistical geometry enables a robust and accurate approach to addressing the problem of structure characterization in water. The snapshot conformations from the molecular simulation of water or aqueous solution by molecular dynamics, Monte Carlo, or other method can be easily tessellated and geometrical parameters of the resulting Delaunay simplices can be measured. Tetrahedrality is a quantitative measure of the degree of distortion of the Delaunay simplices from the regular tetrahedra, that was introduced by Medvedev and Naberukhin (1987) for simple liquids, but can be easily extended to water and other systems. Tetrahedrality is calculated as:

$$T = \sum_{i>j} (l_i - l_j)^2 / 15\bar{l}^2, \quad (3.1)$$

where l_i is the length of the i -th edge, and \bar{l} is the mean length of the edges of the given simplex. For a regular tetrahedron with four equilateral triangular faces, $T = 0$. For any irregular tetrahedron, $T > 0$. In case of a simulated molecular system the tessellation produces a large number of Delaunay simplices for each snapshot conformation, and a number of such conformations can be very large as well. If the simulated system is at equilibrium, the ergodic theorem applies, and time averages along a system trajectory can be combined with ensemble averages over the phase space. Such a combination increases the number of simplices for the analysis by several orders of magnitude ($10^3 - 10^4$ simplices in a conformation multiplied by $10^3 - 10^4$ conformations), which affords high statistical reliability of the results. The nature of this descriptor allows to calculate it separately in confined or limited regions of the simulation system, e.g., in concentric spherical layers around a solute.

The distribution of water tetrahedrality in different layers around solutes depend on the nature of the solute. In the case of charged ions, like ammonium, the difference between tetrahedrality of bulk water and the ammonium hydration water is particularly strong due to the strong hydrogen bonding between water and solute. The peak of the distribution of the tetrahedrality of the ammonium hydration water is shifted to the right which indicates that the hydration water is less tetrahedral than bulk water (Fig. 3.2). Conversely, water in the first hydration shell of methane is just slightly more tetrahedral than the bulk water. Thus,

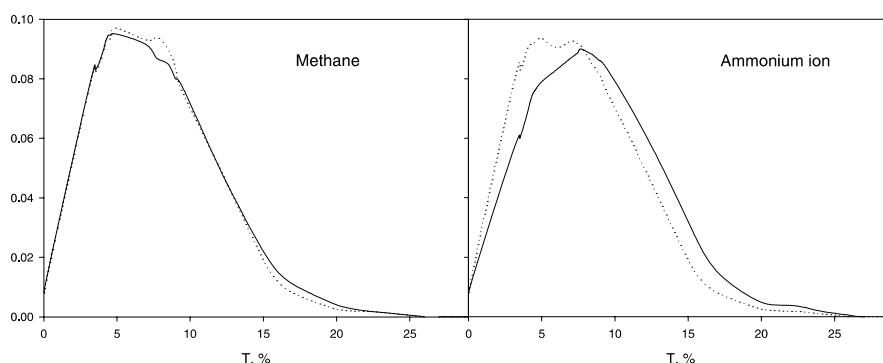


Figure 3.2. Distribution of tetrahedrality of water around solutes; *solid line* – first hydration shell, *dotted line* – bulk water

the hydration water around ammonium is significantly more distorted than that around methane as one could expect in the case of hydrophilic and hydrophobic hydration, respectively (Vaisman et al., 1994).

It is worth to note that the presence of hydrophobic solute changes the distribution of water tetrahedrality in the same way as the decrease of temperature. This observation is consistent with the concept of the decrease of ‘structural temperature’ of water, surrounding hydrophobic particles, that has been discussed in the literature for a long time. The decrease in the structural temperature corresponds to the increased structural order of water, because any structural characteristic of liquid must be a monotonically decreasing function of temperature. Distribution of tetrahedrality entirely complies with this requirement.

The influence of both solutes on the water tetrahedrality is almost unobservable beyond the first hydration shell. The distribution of tetrahedrality for both solutions is similar at both cutoff radii (Vaisman et al., 1994). This result indicates that the distribution of tetrahedrality is not sensitive to the treatment of long-range interactions. Distribution of tetrahedrality beyond the first hydration shell is similar to that in pure water.

Spatial and Compositional Three-dimensional Patterns in Proteins

3.4

Delaunay simplices obtained as a result of the tessellation can be used to define objectively the nearest neighbor residues in 3D protein structures. The most significant feature of Delaunay tessellation, as compared with other methods of nearest neighbor identification, is that the number of nearest neighbors in three dimensions is always four, which represents a fundamental topological property of 3D space. Statistical analysis of the amino acid composition of Delaunay sim-

plices provides information about spatial propensities of all quadruplets of amino acid residues clustered together in folded protein structures. The compositional statistics can be also used to construct four-body empirical contact potentials, which may provide improvement over traditional pairwise statistical potentials (e.g., Miyazawa and Jernigan, 2000) for protein structure analysis and prediction.

To perform the tessellation protein residues should be represented by single points located, for example, in the positions of the C_{α} atoms or the centers of the side chains. Tessellation training set includes high-quality representative protein structures with low primary-sequence identity (Wang and Dunbrack, 2003). The tessellated proteins are analyzed by computing various geometrical properties and compositional statistics of Delaunay simplices.

An example of Delaunay tessellation of a folded protein is illustrated on Fig. 3.3 for crambin (1 crn). The tessellation of this 46-residue protein generates an aggregate of 192 nonoverlapping, space-filling irregular tetrahedra (Delaunay simplices). Each Delaunay simplex uniquely defines four nearest neighbor C_{α} atoms and thus four nearest neighbor amino acid residues.

For the analysis of correlations between the structure and sequence of proteins, we introduced a classification of simplices based on the relative positions of vertex residues in the primary sequence (Singh et al., 1996). Two residues were defined as distant if they were separated by one or more residues in the protein primary sequence. Simplices were divided into five nonredundant classes: class {4}, where all four residues in the simplex are consecutive in the protein primary sequence; class {3, 1}, where three residues are consecutive and the fourth is a distant one; class {2, 2}, where two pairs of consecutive residues are separated in the sequence; class {2, 1, 1}, where two residues are consecutive, and the other two are distant both from the first two and from each other; and class {1, 1, 1, 1} where all four residues are distant from each other (Fig. 3.4). All five classes usually occur in any given protein.

The differences between classes of simplices can be evaluated using geometrical parameters of tetrahedra such as volume and tetrahedrality (3.1). Distributions of

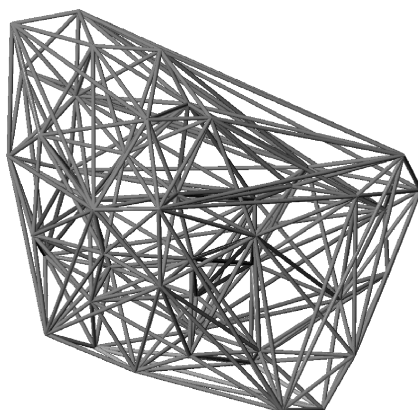


Figure 3.3. Delaunay tessellation of Crambin

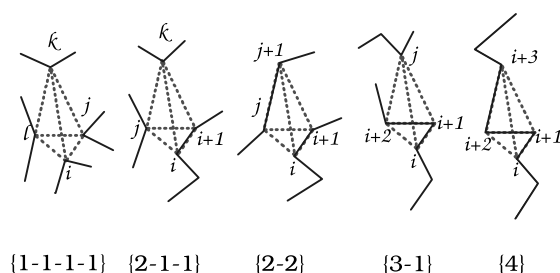


Figure 3.4. Five classes of Delaunay simplices

volume and tetrahedrality for all five classes of simplices is shown in Fig. 3.5. The sharp narrow peaks correspond to the simplices of classes {4} and {2, 2}. They tend to have well defined distributions of volume and distortion of tetrahedrality. These results suggest that tetrahedra of these two classes may occur in regular protein conformations such as α -helices and may be indicative of a protein fold family. We have calculated the relative frequency of occurrence of tetrahedra of each class in each protein in a small dataset of hundred proteins from different families and plotted the results in Fig. 3.6. The proteins were sorted in the ascending order of fraction of tetrahedra of class {4}. Noticeably, the content of simplices of class {3, 1} decreases with the increase of the content of class {4} simplices. According to common classifications of protein fold families (Orengo et al., 1997), at the top level of hierarchy most proteins can be characterized as all-alpha, all-beta, or alpha/beta. The fold families for the proteins in the dataset are also shown in Fig. 3.6. These results suggest that proteins having a high content of tetrahedra of classes {4} and {2, 2} (i.e., proteins in the right part of the plot in Fig. 3.6) belong to the family of all-alpha proteins. Similarly, proteins having a low content of tetrahedra of classes {4} and {2, 2} but a high content of tetrahedra of classes {2, 2} and {3, 1} (i.e., proteins in the left part of the plot in Fig. 3.6) belong to the all-beta protein fold family. Finally, proteins in the middle of the plot belong to the alpha/beta fold family. Thus, the results of this analysis show that the ratio of tetrahedra of different classes is indicative of the protein fold family.

Identification of significant patterns in biomolecular objects depends on the possibility to distinguish what is likely from what is unlikely to occur by chance (Karlín et al., 1991). Statistical analysis of amino acid composition of the Delaunay simplices provides information about spatial propensities of all quadruplets of amino acid residues to be clustered together in folded protein structures. We analyzed the results of the Delaunay tessellation of these proteins in terms of statistical likelihood of occurrence of four nearest neighbor amino acid residues for all observed quadruplet combinations of 20 natural amino acids. The log-likelihood factor, q , for each quadruplet was calculated using the following equation:

$$q_{ijkl} = \log \frac{f_{ijkl}}{p_{ijkl}} \quad (3.2)$$

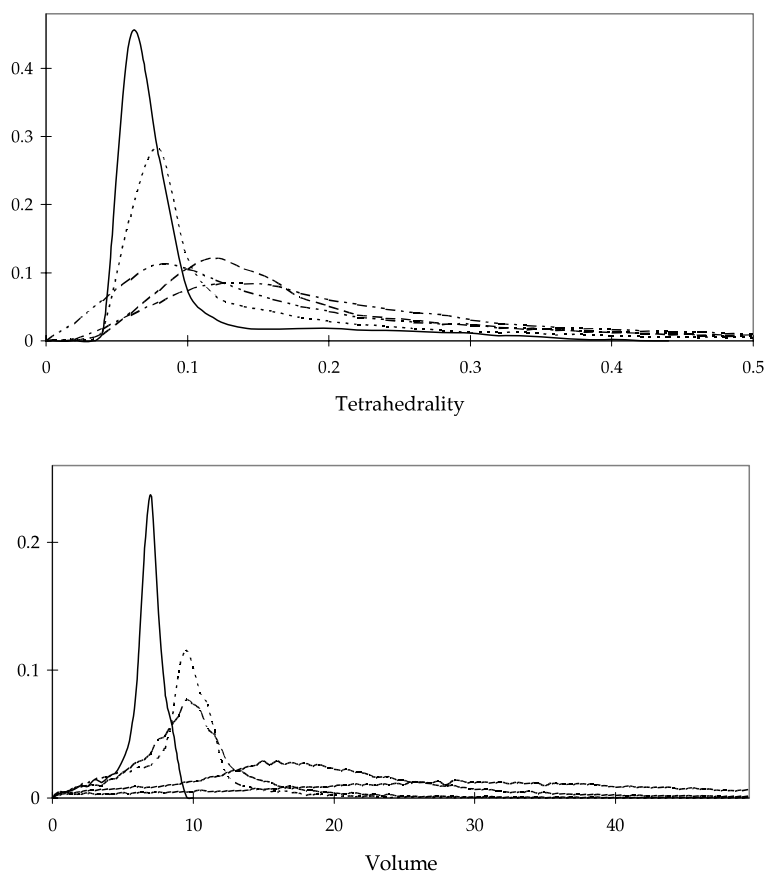


Figure 3.5. Distribution of tetrahedrality and volume (in \AA^3) of Delaunay simplices in proteins

where i, j, k, l are any of the 20 natural amino acid residues, f_{ijkl} is the observed normalized frequency of occurrence of a given quadruplet, and p_{ijkl} is the randomly expected frequency of occurrence of a given quadruplet. The q_{ijkl} shows the likelihood of finding four particular residues in one simplex. The f_{ijkl} is calculated by dividing the total number of occurrence of each quadruplet type by the total number of observed quadruplets of all types. The p_{ijkl} was calculated from the following equation:

$$p_{ijkl} = C a_i a_j a_k a_l \quad (3.3)$$

where $a_i, a_j, a_k,$ and a_l are the observed frequencies of occurrence of individual amino acid residue (i.e. total number of occurrences of each residue type divided by the total number of amino acid residues in the dataset), and C is the permutation factor, defined as

$$C = \frac{4!}{\prod_i^n (t_i!)} \quad (3.4)$$

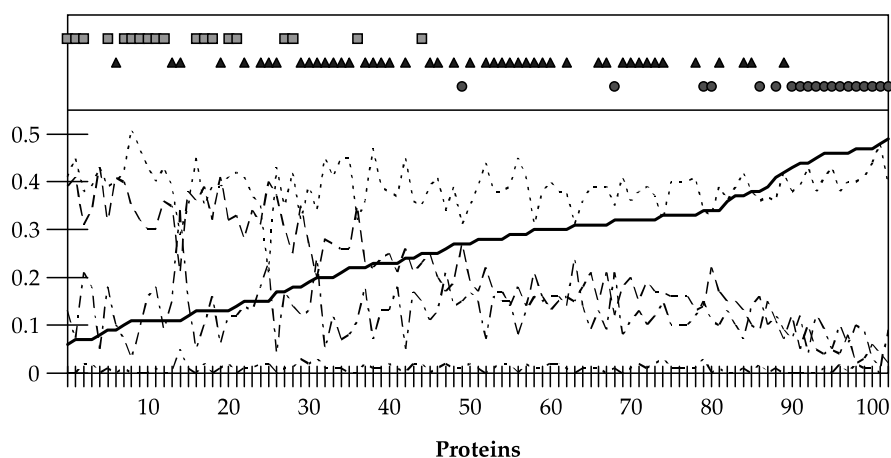


Figure 3.6. Classes of Delaunay simplices and protein fold families. Contents of simplices of class {4} (solid line), class {3, 1} (dashed line), class {2, 1} (dotted line), class {2, 1} (dash-dotted line), class {1, 1, 1, 1} (dash-dot-dotted line). Upper part of the figure displays fold family assignment: all-alpha (circles), all-beta (squares), and alpha-beta (triangles)

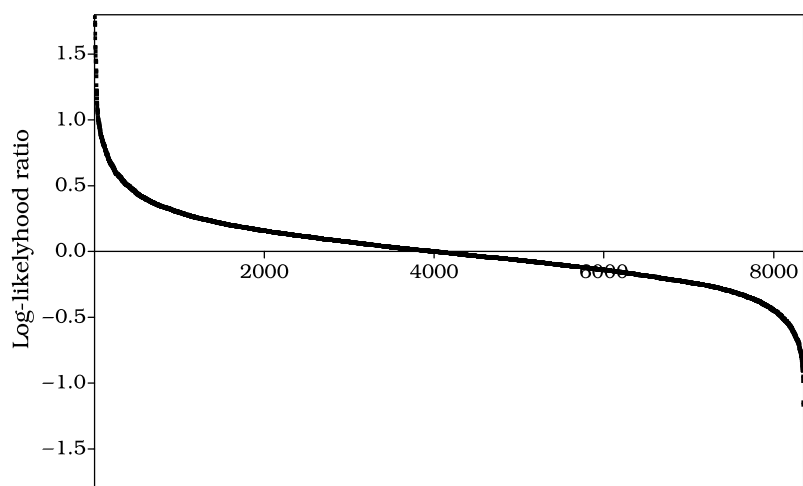
where n is the number of distinct residue types in a quadruplet and t_i is the number of amino acids of type i . The factor C accounts for the permutability of replicated residue types.

Theoretically, the maximum number of all possible quadruplets of 20 natural amino acid residues is 8855 ($C_{20}^4 + 3C_{20}^3 + 2C_{20}^2 + C_{20}^2 + C_{20}^1$). The first term accounts for simplices with four distinct residue types, the second – three types in 1 – 1 – 2 distribution, the third – two types in 1 – 3 distribution, the fourth – two types in 2 – 2 distribution, and the fifth – four identical residues. The log-likelihood factor q is plotted in Fig. 3.7 for all observed quadruplets of natural amino acids. Each quadruplet is thus characterized by a certain value of the q factor which describes the nonrandom bias for the four amino acid residues to be found in the same Delaunay simplex. This value can be interpreted as a four-body statistical potential energy function. The statistical potential can be used in a variety of structure prediction, protein modeling, and computational mutagenesis applications.

Computational mutagenesis is based on the analysis of a protein potential profile, which is constructed by summing the log-likelihood scores from (3.2) for all simplices in which a particular residue participates. A plot of the potential profile for a small protein, HIV-1 protease, is shown in Fig. 3.8. The shape of the potential profile frequently reflects important features of the protein, for example, the residues in local maxima values of the profile are usually located in the hydrophobic core of the protein and these residues play an important role in maintaining protein stability.

A potential profile can be easily calculated for both wild type and mutant proteins, assuming that the structural differences between them are small and that their tessellation results are similar. In this case the difference between the

profiles is defined only by the change in composition of the simplices involving the substitution site. The resulting difference profile provides important insights into the changes in protein energetics due to the mutation.



Delaunay simplices with distinct composition

Figure 3.7. Log-likelihood ratio for the Delaunay simplices

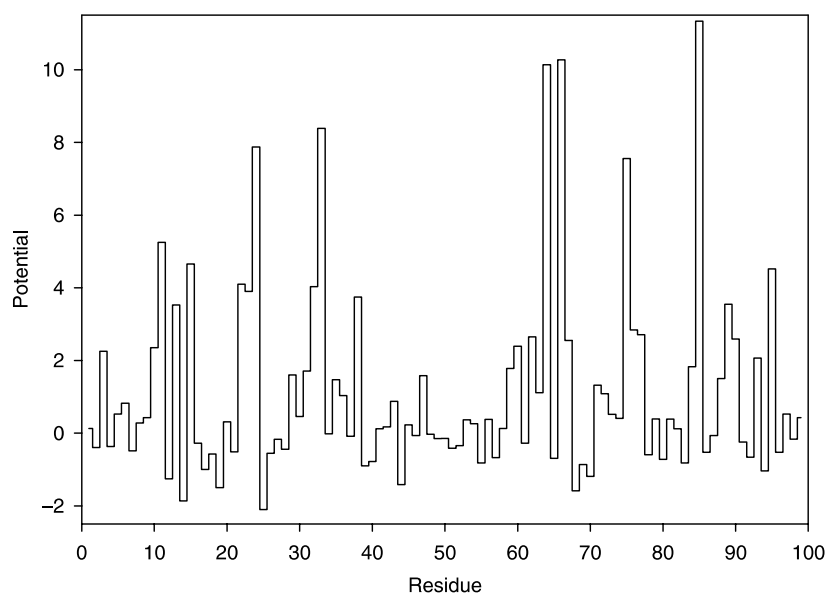


Figure 3.8. Potential profile of HIV-1 protease

Protein Structure Comparison and Classification

3.5

Using the information from the Delaunay tessellation of a protein's backbone, it is possible to build a statistical representation of that protein, which takes into account the way its sequence must "twist and turn" in order to bring each four-body residue cluster into contact. Each residue i, j, k , and l of a four-body cluster comprising a simplex are nearest neighbors in Euclidean space as defined by the tessellation, but are separated by the three distances d_{ij} , d_{jk} , and d_{kl} in sequence space. Based on this idea, we build a 1000-tuple representation of a single protein by making use of two metrics: (1) the Euclidean metric used to define the Delaunay tessellation of the protein's C_α atomic coordinates and (2) the distance between residues in sequence space.

If we consider a tessellated protein with N residues integrally enumerated according to their position along the primary sequence, the length of a simplex edge in sequence space can be defined as $d_{ij} = j - i - 1$, where d_{ij} is the length of the simplex edge, \overline{ij} , corresponding to the i th and j th α -carbons along the sequence. If one considers the graph formed by the union of the simplex edge between the two points i and j and the set of edges between all d_{ij} points along the sequence between i and j , it is seen that the Euclidean simplex edge, \overline{ij} , can generally be classified as a far edge (Pandit and Amritkar, 1999). Every simplex in the protein's tessellation will have three such edges associated with its vertices: i, j, k , and l where i, j, k , and l are integers corresponding to C_α atoms enumerated according to their position along the primary sequence. Thus, we proceed to quantify the degree of "farness" in an intuitive way, by applying a transformation, T , which maps the length, d , of each edge to an integer value according to

$$T : d \mapsto \begin{cases} 1 & \text{if } d = 0 \\ 2 & \text{if } d = 1 \\ 3 & \text{if } d = 2 \\ 4 & \text{if } d = 3 \\ 5 & \text{if } 4 \leq d \leq 6 \\ 6 & \text{if } 7 \leq d \leq 11 \\ 7 & \text{if } 12 \leq d \leq 20 \\ 8 & \text{if } 21 \leq d \leq 49 \\ 9 & \text{if } 50 \leq d \leq 100 \\ 10 & \text{if } d \geq 101 \end{cases} \quad (3.5)$$

The reasoning behind the design of the transformation is described by Bostick and Vaisman (2003). This transformation is used to construct an array that is representative of the distribution of combinations of segment lengths along the protein backbone forming four-residue clusters within the protein's structure as defined by the tessellation of its C_α atomic coordinates. Each simplex in the protein's

tessellation contributes to a 3D array, M , where M_{npr} is the number of simplices whose edges satisfy the following conditions:

1. The Euclidean length of any one simplex edge is not greater than 10 \AA .
2. $d_{ij} = n$
3. $d_{jk} = p$
4. $d_{kl} = r$

Condition 1 is provided because simplices with a Euclidean edge length above 10 \AA are generally a result of the positions of α -carbons on the exterior of the protein. We filter out contributions from these simplices to M , because they do not represent physical interactions between the participating residues. The simplices with the long edges are formed due to the absence of solvent and other molecules around the protein in the tessellation, they would not have existed if the protein was solvated. The data structure, M , contains 1000 elements. The number of elements is invariant with respect to the number of residues of the protein. In order to more easily conceptualize the mapping of the protein topology to the data structure, M , we rewrite it as a 1000-tuple vector $\vec{M} = \{M_{000}, M_{001}, \dots, M_{010}, M_{011}, \dots, M_{999}\}$.

Given that each element of this vector represents a statistical contribution to the global topology, a comparison of two proteins making use of this mapping must involve the evaluation of the differences in single corresponding elements of the proteins' 1000-tuples. We define, therefore, a raw topological score, Q , representative of the topological distance between any two proteins represented by data structures, \vec{M} and \vec{M}' , as the supremum norm,

$$Q \equiv \left\| \vec{M} - \vec{M}' \right\|_{\text{sup}} \equiv \sum_{i=0}^{999} |M_i - M'_i|. \quad (3.6)$$

This norm is topologically equivalent to the Euclidean norm and has the added advantage that it is less computationally expensive to calculate.

This topological score has an obvious dependence on the sequence length difference between the two proteins being compared due to the following implicit relation for a single protein representation,

$$N_s = \sum_{i=0}^{999} M_i, \quad (3.7)$$

where i is the number of simplices with no edge having a Euclidian length greater than 10 \AA , and the M_i are the elements of the protein representation. In other words, since N_s is proportional to the number of residues in the protein, the difference in the length between two compared proteins might provide a systematic extraneous contribution to their score, Q , in (3.6). This is not to say that the sequence length of a protein does not play a role in its topology. In fact, the length should be quite crucial (Bostick et al, 2004). However, the length dependence of our score implied by (3.7) is endemic to our protein representation (derived from its tessellation),

and not due to protein topology itself. This length dependence may be removed by first normalizing the vector representation as follows:

$$\underline{M} = \frac{\vec{M}}{\|\vec{M}\|} \quad (3.8)$$

resulting in the unit-vector representation, \underline{M} . The corresponding normalized topological score,

$$\underline{Q} \equiv \|\underline{M} - \underline{M}\|_{\text{sup}} \quad (3.9)$$

can be expected to be less sensitive to the chain length difference between the two proteins being compared. Despite normalization, however, this score should still have an inherent dependence on the length difference between the compared proteins. A protein's structure must be dependent on the length of its sequence, because the number of configurational degrees of freedom in a polymer's structure is proportional to the number of residues it possesses. Such a dependence on the size of compared proteins is present in geometric methods of comparison such as structural alignment as well, and in some cases, has been accounted for (Carugo and Pongor, 2001).

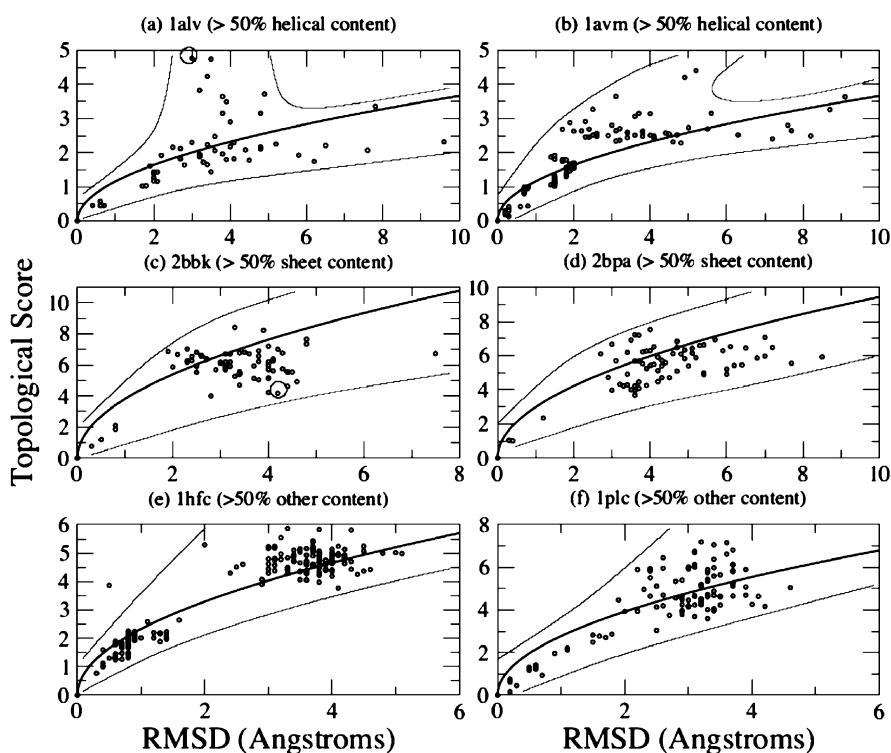


Figure 3.9. Topological and geometric comparison within 6 protein families

The results of topological protein structure comparison can be illustrated using an example of proteins that belong to the same family. Six protein families were selected from the FSSP (Families of Structurally Similar Proteins) database for topological evaluation. We selected families that span various levels of secondary structural content. The representatives of these families are as follows: 1alv and 1avm (having greater than 50% α -helical content), 2bbk and 2bpa (having greater than 50% β -sheet content), and 1hfc and 1plc (having at least 50% content that is classified as neither α -helical nor β -sheet). The FSSP database contains the results of the alignments of the extended family of each of these representative chains. Each family in the database consists of all structural neighbors excluding very close homologs (proteins having a sequence identity greater than 70%). The topological score was calculated for each representative in a one-against-all comparison with its neighbors. All of the scores are plotted against RMSD for each of the families in Fig. 3.9. A strong correlation between the topological score and structure similarity and the power-law trend can be seen for all families.

3.6

Conclusions

Methods of statistical and computational geometry, Voronoi and Delaunay tessellation in particular, play an increasingly important role in exploration of complex nature of molecular and biomolecular structure. The range of applications of statistical geometry for biomolecular structural studies has grown significantly in the past decade. As the new experimental structural information on biomolecules becomes available, the need for sophisticated and robust tools for its interpretation will further increase. At the same time more known structure will enable the creation of larger and better training sets for pattern identification. Existing and new statistical geometry algorithms may prove instrumental in future developments of methods for protein structure analysis, ab initio structure prediction, and protein engineering.

References

- Angelov, B., Sadoc, J.F., Jullien, R., Soyer, A., Mornon, J.P. and Chomilier, J. (2002). Nonatomic solvent-driven Voronoi tessellation of proteins: an open tool to analyze protein folds. *Proteins*, 49(4)446–456.
- Aurenhammer, F. (1991). Voronoi diagrams – a survey of a fundamental geometric data structure. *ACM Computing Surveys*, 23: 345–405.
- Bakowies, D. and van Gunsteren, W.F. (2002). Water in protein cavities: A procedure to identify internal water and exchange pathways and application to fatty acid-binding protein. *Proteins*, 47(4)534–545.
- Barber, C.B., Dobkin, D.P. and Huhdanpaa, H.T. (1996). The Quickhull algorithm for convex hulls. *ACM Trans on Mathematical Software*, 22(4)469–483.

- Bernal, J.D. (1959). A geometrical approach to the structure of liquids. *Nature*, 183(4655)141–147.
- Bostick D. and Vaisman, I.I. (2003). A new topological method to measure protein structure similarity. *Biochem Biophys Res Commun*, 304(2)320–325.
- Bostick, D., Shen, M. and Vaisman, I.I. (2004). A simple topological representation of protein structure: Implications for new, fast, and robust structural classification. *Proteins*, 55.
- Carter, C.W. Jr, LeFebvre, B.C., Cammer, S.A., Tropsha, A. and Edgell, M.H. (2001). Four-body potentials reveal protein-specific correlations to stability changes caused by hydrophobic core mutations. *J Mol Biol*, 311(4)625–638.
- Carugo, O. and Pongor, S. (2001). A normalized root-mean-square distance for comparing protein three-dimensional structures. *Protein Sci*, 10(7)1470–1473.
- Chakravarty, S., Bhinge, A. and Varadarajan, R. (2002). A procedure for detection and quantitation of cavity volumes proteins. Application to measure the strength of the hydrophobic driving force in protein folding. *J Biol Chem*, 277(35)31345–31353.
- Chothia, C. (1975). Structural invariants in protein folding. *Nature* 1975, 254(5498)304–308.
- Delaunay, B. N. (1934). Sur la sphere vide, *Izv Akad Nauk SSSR, Otd Mat Est Nauk*, 7:793–800.
- Edelsbrunner, H., Kirkpatrick, D.G. and Seidel, R. (1983). On the shape of a set of points in the plane. *IEEE Transactions on Information Theory*, IT-29(4)551–559.
- Finney, J.L. (1970). Random packing and the structure of simple liquids. I. The geometry of random close packing. *Proc Roy Soc Lond*, A319479–493 and 495–507.
- Finney, J.L. (1975). Volume occupation, environment and accessibility in proteins. The problem of the protein surface. *J Mol Biol*, 96(4)721–732.
- Gan, H.H., Tropsha, A and Schlick, T. (2001) Lattice protein folding with two and four-body statistical potentials. *Proteins*, 43(2)161–174.
- Gellatly, B.J. and Finney, J.L. (1982). Calculation of protein volumes: an alternative to the Voronoi procedure. *J Mol Biol*, 161(2)305–322.
- Gerstein, M., Tsai, J. and Levitt, M. (1995). The volume of atoms on the protein surface: calculated from simulation, using Voronoi polyhedra. *J Mol Biol*, 249(5)955–966.
- Goede, A., Preissner, R. and Frömmel, C. (1997). Voronoi cell: New method for allocation of space among atoms: Elimination of avoidable errors in calculation of atomic volume and density. *J Comput Chem*, 18(9)1113–1123.
- Harpaz, Y., Gerstein, M. and Chothia, C. (1994). Volume changes on protein folding. *Structure*, 2(7)641–649.
- Karlin, S., Bucher, P. and Brendel, V. (1991). Altschul S.F., Statistical methods and insights for protein and DNA sequences. *Annu Rev Biophys Biophys Chem*, 20:175–203.

- Kobayashi, N. and Go N. (1997). A method to search for similar protein local structures at ligand binding sites and its application to adenine recognition. *Eur Biophys J*, 26(2)135–144.
- Kobayashi, N., Yamato, T. and Go N. (1997). Mechanical property of a TIM-barrel protein. *Proteins*, 28(1)109–116.
- Krishnamoorthy, B. and Tropsha, A. (2003). Development of a four-body statistical pseudo-potential to discriminate native from non-native protein conformations. *Bioinformatics*, 19(12)1540–1548.
- Liang, J., Edelsbrunner, H., Fu, P., Sudhakar, P.V. and Subramaniam, S. (1998a). Analytical shape computation of macromolecules: I. Molecular area and volume through alpha shape. *Proteins*, 33(1)1–17.
- Liang, J., Edelsbrunner, H., Fu, P., Sudhakar, P.V. and Subramaniam, S. (1998b). Analytical shape computation of macromolecules: II. Inaccessible cavities in proteins. *Proteins*, 33(1)18–29.
- Liang, J., Edelsbrunner, H., Woodward, C. (1998c). Anatomy of protein pockets and cavities: measurement of binding site geometry and implications for ligand design. *Protein Sci*, 7(9)1884–1897.
- Lo Conte, L., Chothia, C. and Janin, J. (1999). The atomic structure of protein-protein recognition sites. *J Mol Biol*, 285(5)2177–2198.
- Masso, M. and Vaisman, I.I. (2003). Comprehensive mutagenesis of HIV-1 protease: a computational geometry approach. *Biochem Biophys Res Commun*, 305(2)322–326.
- Medvedev, N.N. and Naberukhin, Yu.I. (1987). Analysis of structure of simple liquids and amorphous solids by statistical geometry method. *Zh Strukt Khimii*, 28(3)117–132.
- Miyazawa, S. and Jernigan, R.L. (2000). Identifying sequence-structure pairs undetected by sequence alignments. *Protein Eng*, 13(7)459–475.
- McConkey, B.J., Sobolev, V. and Edelman, M. (2002). Quantification of protein surfaces, volumes and atom-atom contacts using a constrained Voronoi procedure. *Bioinformatics*, 18(10)1365–1373.
- Munson, P.J. and Singh, R.K. (1997). Statistical significance of hierarchical multi-body potentials based on Delaunay tessellation and their application in sequence-structure alignment. *Protein Sci*, 6(7)1467–1481.
- Orengo, C.A., Michie, A.D., Jones, S., Jones, D.T., Swindells, M.B. and Thornton, J.M. (1997). CATH – a hierarchic classification of protein domain structures. *Structure*, 5(8)1093–1108.
- Pandit, S. A. and Amritkar, R. E. (1999). Characterization and control of small-world networks. *Phys Rev E*, 60:1119–1122.
- Pontius, J., Richelle, J. and Wodak, S.J. (1996). Deviations from standard atomic volumes as a quality measure for protein crystal structures. *J. Mol. Biol.*, 264(1)121–136.
- Quillin, M.L. and Matthews, B.W. (2000). Accurate calculation of the density of proteins. *Acta Crystallogr D Biol Crystallogr*, 56(Pt 7)791–794.
- Richards, F.M. (1974). The interpretation of protein structures: total volume, group volume distributions and packing density. *J Mol Biol*, 82(1)1–14.

- Richards, F.M. (1985). Calculation of molecular volumes and areas for structures of known geometry. *Methods Enzymol*, 115:440–464.
- Rother, K, Preissner, R, Goede, A and Frömmel, C. (2003). Inhomogeneous molecular density: reference packing densities and distribution of cavities within proteins. *Bioinformatics*, 19(16)2112–2121.
- Sadoc, J.F., Jullien, R. and Rivier, N. (2003). The Laguerre polyhedral decomposition: application to protein folds. *Eur Phys J B*, 33(3)355–363.
- Schaefer, M., Bartels, C., Leclerc, F. and Karplus M. (2001). Effective atom volumes for implicit solvent models: comparison between Voronoi volumes and minimum fluctuation volumes. *J Comput Chem*, 22(15)1857–1879.
- Singh, R.K., Tropsha, A. and Vaisman, I.I. (1996). Delaunay tessellation of proteins: four body nearest-neighbor propensities of amino acid residues. *J Comput Biol*, 3(2)213–222.
- Soyer, A., Chomilier, J., Moron, J.P., Jullien, R. and Sadoc, J.F. (2000). Voronoi tessellation reveals the condensed matter character of folded proteins. *Phys Rev Lett*, 85(16)3532–3535.
- Sugihara, K. and Inagaki, H. (1995). Why is the 3D Delaunay triangulation difficult to construct? *Information Processing Letters*, 54:275–280.
- Tropsha, A., Singh, R.K., Vaisman, I.I. and Zheng, W. (1996). Statistical geometry analysis of proteins: implications for inverted structure prediction. *Pac Symp Biocomput*, 614–623.
- Tropsha, A., Carter, C.W. Jr, Cammer, S. and Vaisman, I.I. (2003). Simplicial neighborhood analysis of protein packing (SNAPP): a computational geometry approach to studying proteins. *Methods Enzymol*, 374:509–544.
- Tsai, J., Taylor, R., Chothia, C. and Gerstein, M. (1999). The packing density in proteins: standard radii and volumes. *J Mol Biol*, 290(1)253–266.
- Tsai, J. and Gerstein, M. (2002). Calculations of protein volumes: sensitivity analysis and parameter database. *Bioinformatics*, 18(7)985–995.
- Naberukhin, Y.I., Voloshin, V.P. and Medvedev, N.N. (1991). Geometrical analysis of the structure of simple liquids: percolation approach. *Mol Physics*, 73:917–936.
- Vaisman, I.I., Perera, L. and Berkowitz, M.L. (1993). Mobility of stretched water. *J Chem Phys*, 98(12)9859–9862.
- Vaisman, I.I. and Berkowitz, M.L. (1992). Local structural order and molecular associations in water-DMSO mixtures. Molecular dynamics study. *J Am Chem Soc*, 114(20)7889–7896.
- Vaisman, I.I., Brown, F.K. and Tropsha, A. (1994). Distance Dependence of Water Structure Around Model Solutes. *J Phys Chem*, 98(21)5559–5564.
- Vaisman, I.I., Tropsha, A. and Zheng W. (1998). Compositional preferences in quadruplets of nearest neighbor residues in protein structures: Statistical geometry analysis. *Proc. of the IEEE Symposia on Intelligence and Systems*, 163–168.
- Voloshin, V.P., Naberukhin, Y.I. and Medvedev, N.N. (1989). Can various classes of atomic configurations (Delaunay simplices) be distinguished in random close packing of spherical particles?, *Molec Simulation*, 4:209–227.

- Voronoi, G.F. (1908). Nouvelles applications des paramètres continus à la théorie des formes quadratiques. Deuxième Mémoire: Recherches sur les parallélogrammes primitifs. *J Reine Angew Math*, 134:198–287.
- Wang, G. and Dunbrack, R.L. Jr. (2003). PISCES: a protein sequence culling server. *Bioinformatics*, 19(12):1589–1591.
- Weberndorfer, G., Hofacker, I.L. and Stadler, P.F. (1999). An efficient potential for protein sequence design. *Proc German Conf Bioinformatics*, 107–112.
- Wernisch, L., Hunting, M. and Wodak, S.J. (1999). Identification of structural domains in proteins by a graph heuristic. *Proteins*, 35(3):338–352.
- Zheng, W., Cho, S.J., Vaisman, I.I. and Tropsha, A. (1997). A new approach to protein fold recognition based on Delaunay tessellation of protein structure. *Pac Symp Biocomput*, 486–497.

Functional Magnetic Resonance Imaging

IV.4

William F. Eddy, Rebecca L. McNamee

4.1	<i>Introduction: Overview and Purpose of fMRI</i>	1002
4.2	<i>Background</i>	1003
	Magnetic Resonance (MR)	1003
	Magnetic Resonance Imaging (MRI)	1004
	Functional MRI	1006
4.3	<i>fMRI Data</i>	1008
	Design of an fMRI Experiment	1008
	Data Collection.....	1010
	Sources of Bias and Variance in the Data.....	1011
4.4	<i>Modeling and Analysis</i>	1013
	An Ideal Model	1013
	A Practical Approach.....	1014
4.5	<i>Computational Issues</i>	1020
	Software Packages for fMRI Data Analysis	1021
	Other Computational Issues	1023
4.6	<i>Conclusions</i>	1023

Introduction: Overview and Purpose of fMRI

4.1

The 2003 Nobel Prize in Medicine went to Paul Lauterbur and Sir Peter Mansfield for the invention of magnetic resonance imaging (MRI) in the 1970 s. Since its invention MRI has rapidly changed the world of medicine; there are currently more than 20,000 MRI scanners in the world and many millions of images are generated by them each year. In the early 1990 s, Ogawa et al. (1992), Belliveau et al. (1991) and Kwong et al. (1992) showed that MRI could be used for the detection of brain function. Because the technique is non-invasive and does not require the injection of dyes or radioactive tracers, functional MRI (fMRI), has opened up opportunities that were never before possible for studying the living human brain in its working state.

One of the primary uses for fMRI is the mapping of brain function onto brain structure. This is done by engaging a subject in a specific motor, sensory, or cognitive task while collecting MR images of the brain. The regions of increased activity are presumed to be those which perform the task. A particular example is given in Fig. 4.1.

Although mapping of function to structure is an important use of fMRI, the possibilities of its application for investigating the dynamics of brain function are many. Researchers have recently begun using fMRI to study brain development in



Figure 4.1. Brain activity while performing a short term memory task, in a high school athlete with mild traumatic brain injury. This single slice shows only a portion of the activity in the entire brain. Because it was derived by thresholding a test statistic there may be both false positive and false negative pixels. The physical contiguity of the regions of activity suggests that there are not any false positives

both normal and pathological situations (Gaillard et al., 2001). The method can also be used to examine the aging brain (Rosen et al., 2002), as well as to study the brain under situations of learning (Poldrack, 2000) and injury (McAllister et al., 1999).

Scientific fields other than psychology and neuroscience are also developing an interest in fMRI research. For example, pharmaceutical applications may use fMRI to investigate the brain before and after the administration of a drug, and geneticists may be interested in how the expression of similar or different genotypes may alter brain functioning in one individual versus another.

As the field of fMRI grows, the problems that it presents for statisticians and other quantitative scientists are also growing. There are several reviews of fMRI work in the statistical literature; see, e.g., Eddy et al. (1999), Lange (1996) and Lazar et al. (2001). While collecting the data from subjects has become easier, the data sets are usually very large (100 MB or more) and are especially variable, containing both systematic and random noise. Storage, processing, and analysis of fMRI data are complicated, and the computational problems are legion. In this chapter a brief background of fMRI is first given from a physics and psychology point of view. A full description of the fMRI data as well as the challenges that it presents from a computational statistics viewpoint are then discussed in detail.

Background

4.2

Magnetic Resonance (MR)

4.2.1

Atomic nuclei spin like gyroscopes. When placed into a magnetic field, atomic nuclei that are affected by magnetism (those having an odd number of protons or neutrons or both) align their axis of rotation with the magnetic field. Like a gyroscope the axis of rotation itself rotates; this rotation is called precession. Each nucleus precesses within the magnetic field. The frequency of this precession, the Larmor frequency, is proportional to the strength of the magnetic field, with the constant of proportionality being determined by the gyromagnetic ratio of the atomic species. The relationship can be described by the Larmor equation

$$\omega_0 = \gamma B_0$$

where ω_0 is the Larmor frequency, γ is the gyromagnetic ratio, and B_0 is the strength of the applied magnetic field.

Hydrogen is currently the most widely used element for MR imaging because of its abundance in biological tissue and the strength of the emitted signal. Hydrogen atoms have a gyromagnetic ratio of approximately 42 MHz per Tesla. A Tesla is a measure of the strength of the magnetic field (B_0) with one Tesla equal to 10,000 gauss. For reference, one-half gauss is roughly the strength of the earth's

magnetic field. An MR scanner that is used for functional imaging has a typical field strength of 3 Tesla. The resulting Larmor frequency is about 126 MHz; for comparison a kitchen microwave oven operates at about 2450 MHz.

Some values of the constants γ for other atomic species can be found at http://www.stat.cmu.edu/~fiasco/index.php?ref=reference/ref_constants.shtml

As the nuclei of the hydrogen atoms precess within the magnetic field, the atoms will either line up along the field or against it (that is, the atoms will line up at either 0 or 180 degrees). The strength of the magnetic field and the energy state of the system affect the number of atoms that line up accordingly. Then, while the atoms precess in a steady-state fashion within the magnetic field, a pulse of energy is injected into the system in the form of a transient radio-frequency (rf) pulse perpendicular to the B_0 field at the Larmor frequency (ω_0). This rf pulse excites the atoms at their resonant frequency, causing them to tilt out of alignment with the magnetic field.

As these excited atoms return to equilibrium within the magnetic field they emit rf energy which is collected by an antenna and receiver. Their return to steady-state is known as relaxation, and the signal that the atoms emit is known as the free-induction decay (FID) signal. The FID signal reflects the distribution of hydrogen atoms in the tissue and is used to construct images (see, e.g., Buxton, 2002).

4.2.2 Magnetic Resonance Imaging (MRI)

As described, the basic theory of MR can be used to create images based on the distribution of hydrogen atoms or protons in the tissue sample. Other types of atoms can also be imaged. In these cases, the applied rf pulse must be applied at the Larmor frequency of the atoms of interest in the tissue sample.

In order to create images using MR, an FID signal must be encoded for each tissue dimension, and certain considerations must be made to recognize spatial information in the tissue sample from which the FID signals are being recorded. As outlined above, the resonant frequency at which the atoms must be excited to flip them is dependent on the magnetic field strength. Thus, by adjusting the magnetic field strength at certain locations in the tissue and sending rf pulses at the corresponding resonant frequency, only atoms at the location of interest will be excited. In this manner, spatial information can be resolved.

To aid in the understanding of this principle, consider slice selection through a sample of tissue. As shown in Fig. 4.2, the object under consideration (in this case a person) is placed with the xy -slice axis perpendicular to the magnetic field. A linear magnetic field gradient is then applied in a direction parallel to the bore of the magnet (z -direction). In this manner, each xy -slice of the tissue is subjected to a slightly different magnetic field strength. If the linear gradient at z is equal to $azB_1 + B_0$, then the Larmor frequency at $z = 1$ becomes ω_1 , where:

$$\omega_1 = \gamma(aB_1 + B_0).$$

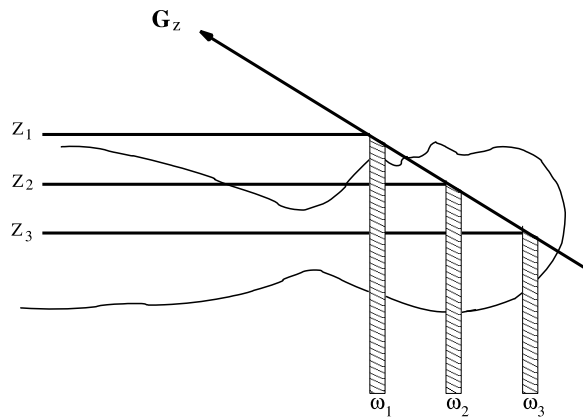


Figure 4.2. This figure (redrawn from Lazar et al., 2001) is a schematic showing how slice selection takes place

The magnetic field strength along each slice of the tissue is now slightly different, so the resonant frequency for each slice of the tissue will be slightly different. By adjusting the rf pulse to correspond to ω_1 , only slices of interest will be excited and imaged. This same principle can be used to define spatial locations in the xy plane as well. The interested reader should refer, for example, to Buxton's 2002 book *Introduction to Functional Magnetic Resonance Imaging: Principles & Techniques*.

The signal intensity from the tissue is a function of the density of hydrogen atoms or protons in the tissue. The more protons in the tissue, the greater the FID signal, and the higher the intensity value. Because different types of tissues vary in their proton density, tissue contrast can be achieved in the images. Contrast also depends on the tissue specific parameters, longitudinal relaxation time (T_1), and transverse relaxation time (T_2). The amount of time that it takes for the longitudinal magnetization of the tissue to return to 63% of its original value after an rf pulse is applied is denoted T_1 , and the time that it takes for the transverse magnetization to return to 37% of its original value after the applied rf pulse is denoted T_2 .

The imaging parameters of TR (time of repetition of the rf pulse) and TE (time of echo before FID signal is measured) can be varied to achieve the maximum contrast for the tissues of interest by considering their T_1 and T_2 properties. This process is also known as weighting or contrasting, and images are usually T_1 -weighted, T_2 -weighted, or proton density weighted. For example, short TRs and TEs lead to T_1 weighted images, because at this time the differences due to the T_1 tissue properties will be the most apparent. In T_1 weighted images, the intensity of bone tissue is typically bright while fluids are dark.

On the other hand, to achieve the best contrast using T_2 properties, a long T_R and a long T_E are used. In T_2 weighted images, bone tissue is dark and fluid areas are light. Proton density weighting is achieved by using a long T_R and a short T_E . With this type of weighting, areas that have the highest proton density are the brightest.

These areas would include the cerebral spinal fluid (CSF) and gray matter. Again, see Buxton (2002) for details; a statistical approach is given in Glad and Sebastiani (1995).

4.2.3 **Functional MRI**

Early Brain Research

The mysteries of the human brain have perplexed researchers for many centuries. Early ideas about brain functioning date at least as far back as the second century to Galen (and even earlier), who associated imagination, intellect, and memory with brain substance. The notion that the brain consisted of functionally discrete areas did not become an accepted idea until the nineteenth century with the work of Franz Joseph Gall (Finger, 1994). Ensuing research involved examining the relationships between the location of brain lesions and deficits and/or changes in behavior as a way to attribute brain function to structure. Although the technique was effective, this method for studying the brain was not without limitations. Since that time, however, the field of neuroscience has grown because of the development of new methods to explore the human brain in its living working state. These new techniques have been given the general term functional neuroimaging.

Functional Neuroimaging

Functional neuroimaging is the term applied to techniques that can map the activity of the living working brain in space and time. Non-invasive approaches to this mapping have included electrophysiological measurements and metabolic measurements. Techniques to measure the electrophysiological signals include the electroencephalogram (EEG) and the magnetoencephalogram (MEG) (National Research Council, 1992). These methods are thought to record (a weighted integral of) the actual neural activity that is taking place in the brain. Although both EEG and MEG have excellent temporal resolution, in their most common form the measured output signals are an integration of activity from many possible unknown sources. Furthermore, for EEGs these integrated signals are only realized after being filtered through layers of blood vessels, fat, and bone. On the other hand, MEG generally only measures the component of the magnetic field which is perpendicular to the surface of the skull. Both methods typically record only a few hundred different locations; a typical functional MRI study measures the signal at 100,000 locations or so. Thus, the spatial resolution of both EEG and MEG is quite poor. Source localization is a research area devoted to trying to map the locations at which these signals originate, but this has proven to be a very difficult task.

Functional neuroimaging measurements also include Positron Emission Tomography (PET) and fMRI. Both of these techniques have good spatial resolution, but unlike EEG and MEG they record responses to changes in blood flow rather than the direct neural activity. Because of this, these techniques have relatively poor temporal resolution.

PET imaging is carried out by labeling molecules of compounds of interest with positron-emitting isotopes. Isotopes that are often used include Carbon-11, Nitrogen-13, or Oxygen-15. These labeled molecules are termed “probes” or “tracers”. The tracers are distributed in the brain according to their delivery, uptake, metabolism, and excretion properties. As these isotopes decay they emit a positron and a neutrino. The neutrino cannot be detected, but each positron collides with an electron and is annihilated. The annihilation converts the electron and positron from mass into energy in the form of gamma rays. These gamma rays are then detected by the scanner. PET can provide excellent measures of metabolic activity of the brain under conditions of normal and abnormal functioning and has therefore been a highly useful tool in studying brain activity. However, one of the main disadvantages of PET is that it requires the injection of ionizing radiation thereby limiting its use for human subject populations (Cherry and Phelps, 1996).

Functional MRI

Functional MRI uses the same principles as MRI, but it is based on the idea that the magnetic state of the hemoglobin in the bloodstream is dependent on whether or not oxygen is bound to it. Deoxygenated blood is paramagnetic, meaning that the unpaired heme groups cause it to have more magnetic susceptibility than it does when oxygen is attached to the heme groups. In fact, the magnetic susceptibility of blood has been shown to vary linearly with blood oxygenation; see, Thulborn et al. (1982), Buxton (2002), Turner (1995), or Weisskoff and Kiihne (1992).

When neurons in the brain are used, their metabolic demands increase. This begins with an increase in local glucose consumption and leads to an increase in local cerebral blood flow. However, for reasons that are unclear, the increase in cerebral blood flow exceeds the increase in metabolic consumption of local oxygen. Therefore the ratio of oxygenated blood to deoxygenated blood increases in the local blood vessels. The change in this ratio of oxygenated blood to deoxygenated blood leads to changes in the local MR signal. Modulations in the MR signal due to this phenomenon can be detected by the scanner and are known as Blood Oxygen Level Dependent (BOLD) contrast. BOLD contrast is currently the main basis of fMRI; see, e.g., Villringer (2000), Logothetis (2002), Ogawa et al. (1990), Buxton (2002), or Turner (1995).

Several informational limitations are imposed by fMRI that should be considered when carrying out a neuroimaging study. Leading these is the fact that fMRI is an indirect measure of brain activity, and its exact physiological mechanism is not known. A model of the interface between actual brain activity and the fMRI signal is shown in Fig. 4.3. Also, the measured activity obtained with fMRI can include many types of local brain cells because the activation that is measured is essentially a combination of all the “brain activity” in the area. Information is thus blurred during fMRI, since the resolution is based on the “smallest measurable vascular unit.” Finally, as mentioned earlier, fMRI has poor temporal resolution; see, e.g., Villringer (2000).



Figure 4.3. Model of brain activity and fMRI data

In spite of these limitations, fMRI has many advantages over previously used methods for studying the brain. fMRI has much better spatial resolution than EEG or MEG. In fact, although the activity is lumped into small regions, these regions can provide accuracy in the range of 1 mm or so. Next, and perhaps most importantly, fMRI does not require the use of ionizing radiation. This allows it to be used experimentally for many different subject types and under many different types of situations. Other benefits include the fact that the data can be collected fairly easily, and analysis tools are becoming more readily available as the field grows.

4.3 **fMRI Data**

4.3.1 **Design of an fMRI Experiment**

There are at least three aspects to the design of an fMRI experiment: (1) the underlying psychological question, (2) the MR physics that will drive the data collection, and (3) traditional statistical design issues. These factors are not exclusive of each other and must be carefully considered prior to the initiation of the study.

The psychological component of design depends on the type of experimental subjects in question as well as the nature and location of the expected response. For example, regions of brain activity could be explored for a single group of subjects, or the location and extent of brain activation could be compared in two different subject groups. The experimental task must also be designed in order to elicit a functional response specific to the area of interest in the brain. A control state (such as a fixation in a visually guided saccade task) is typically alternated with a functional state (the saccade) in order to compare the two states and find the differentially active brain areas.

The MR physics component depends on the nature of the psychological stimuli, the particular MR scanner being used, and the location of the expected response. Several scanning parameters, such as the number and orientation of slices to be collected, the echo time, T_E , the time of repetition, T_R , the flip angle, and others, must be first chosen. The physics of the scan will depend on these chosen parameters. The physical method for collecting each slice of data, termed the pulse sequence, must also be selected. The pulse sequence is a small program run in the

scanner to manipulate the magnetic and rf fields, and is thus dependent on the type of MR scanner and the pulse sequences available for use.

The statistical aspects of design will help to determine how the data will be analyzed after collection. For example, as mentioned above, the experimental design is often set so that the task state is alternated with a control state. The two conditions can then be statistically compared using methods which rely on hypothesis testing (such as *t*-tests). This type of experimental design is called a block design. The block design is robust in that many repetitions of the two conditions can be carried out, and the experimenter can then average all trials for each voxel. Although this technique can find spatial areas of activation, it has the disadvantage of losing most temporal aspects of the data.

In order to capture the time-dependent features of the data as well as the spatial aspects, single trial fMRI experiments have recently become popular. These experiments examine fMRI signal changes over time as the task is being performed (on a voxel-by-voxel basis) rather than relying on the averaging of large time blocks. The results from these single trial experiments are generally not as robust as those for block designs. Furthermore, since fMRI data is typically very noisy, the data must be pre-processed prior to analysis. The techniques for analyzing single trial fMRI data often model those used for processing evoked potentials EEG data (such as filtering or time-averaging of trials). Because of this, single trial fMRI experiments have often been mislabeled as “event-related” fMRI experiment. Figure 4.4 shows the temporal BOLD response that is generally expected from these types of experiments.

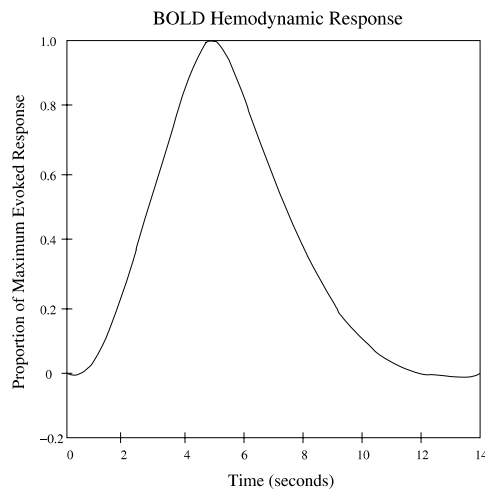


Figure 4.4. BOLD hemodynamic response curve showing the expected contrast changes elicited from a single event. Because the contrast changes are actually due to blood flow changes rather than a direct measure of neural activity, the time scale over which they occur is relatively slow

4.3.2 Data Collection

The raw data from an MR scanner is spatial frequency data. That is, the data are the coefficients of the Fourier representation of the object being imaged. Alternatively we can say that the data are the inverse Fourier transform of the object. The spatial frequency domain has been called “Fourier Space”, “frequency space”, or most popularly “k-space”. The process of taking the inverse Fourier transform to obtain the image has been termed “data reconstruction”. Letting \mathcal{F} be the Fourier transform, we have for an $n \times m$ pixel image I ,

$$\mathcal{F}(I) = \hat{I}(k_x, k_y) = n^{-1}m^{-1} \sum_x^n \sum_y^m I(x, y) \exp(-i2\pi(xk_x + yk_y))$$

In k-space, the low frequencies are located in the center of the image with the frequencies increasing outward with distance from the origin. In a typical k-space plot (see Fig. 4.5), the bulky features of the image lie in the lower frequencies of k-space while the higher frequencies contain the details of the image. In fMRI both the low and high frequency information are important, and the pulse sequence should be designed accordingly.

A typical fMRI data set might consist of a 128 by 128 array of 16 bit complex values recorded for each of 32 two-dimensional slices at each of 450 time points spaced about 1.5 seconds apart. This yields a data set of $2 \times 2 \times 128 \times 128 \times 32 \times 450 = 943,718,400$ bytes, that is approximately 1 gigabyte of data collected in less than 12 minutes. If many experiments are performed on a single subject within the period of an hour or so, and several subjects are examined over time, the necessary storage requirements can become quite extensive. In one of our current studies, we

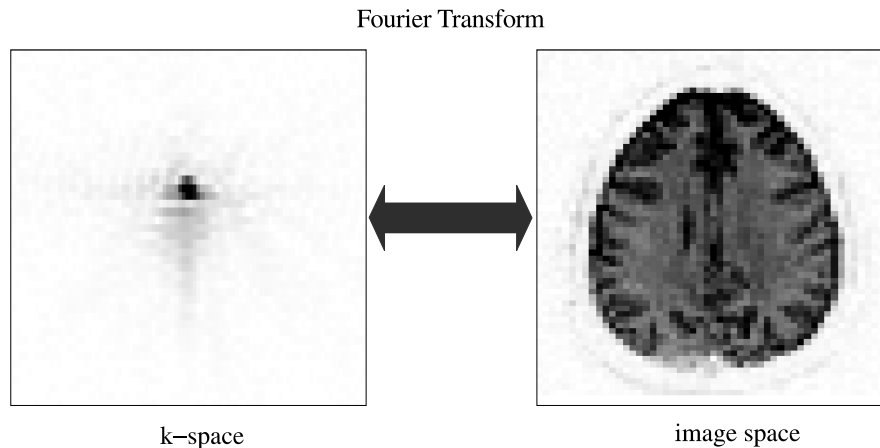


Figure 4.5. Collected fMRI data. The plot on the *left* shows the modulus of the k-space data, and the plot on the *right* shows the modulus of the image. Darker pixels indicate larger values (the opposite of the “radiological convention” derived from X-ray images on photographic film)

anticipate collecting a total of about 700 GB of data. To help deal with this quantity, offline data storage systems are useful. For example, optical disks, CDs, or DVDs can be used to store large amounts of data with minimal effort.

Sources of Bias and Variance in the Data

4.3.3

Areas of brain activity that are found due to specific tasks are dependent on the image to image changes in the measurements within a voxel. Therefore, to produce valid results these changes must be specifically attributable to functional differences in the brain elicited by the performed tasks. Unfortunately, fMRI data is beset with many sources of bias and variability, which can lead to erroneous regions of brain activity and false conclusions about the study. Problems in the data can arise from many sources including the MR scanner itself, the experimental subject, and external interference. Each of these will be discussed with a brief description of the errors that they introduce into the data. The sources of noise in fMRI data can be quite extensive. Although many are covered here, this summary is not exhaustive.

Noise from the Equipment

One main source of bias and systematic variation in fMRI data arises from the MR scanner. The performance of an MR scanner can vary, which can introduce fluctuations in the data, even when the stability measures are well within the instrumental norms (Weisskoff, 1996). Noise from the equipment can occur as systematic or random errors.

Sources of systematic error in the data from the equipment include DC shifts and Nyquist ghosts. DC shifts are also known as baseline errors. This source of data bias is caused by the miscalibration of the analog-to-digital (A/D) converter; the baseline value is not reported as zero. Nyquist ghosts, which are present only in echo-planar imaging, also produce systematic bias in the data. Echo-planar pulse sequences traverse k -space on a boustrophedonic path (back-and-forth as the ox plows the field). Nyquist ghosts are introduced through the mistiming of the oscillating magnetic gradients. The exact time at which the gradient crosses zero is incorrect. This timing error causes an aliasing effect in the reconstructed image and is most prominent in the phase-encode or y direction of the fMRI scan (leading to a ghost of the image repeated at the top and bottom of the true image). Both DC shift errors and Nyquist ghosts that are present in the fMRI data can be corrected to a reasonable extent.

Random errors from the equipment can also cause introduce problems in the fMRI data. One source of unpredictable instability results from inhomogeneities in the static magnetic field of the equipment. Magnetic field inhomogeneities have been reported as one of the most prominent sources of distortion in fMRI studies (Jezzard, 1999). Local variations in the static magnetic field during fMRI will lead to blurring and pixel shifts, which can introduce gross geometric distortions in the images. This problem is especially prominent at regional boundaries in the

sample containing different magnetic susceptibility properties, for example, air-tissue interfaces around the frontal lobes and bone-tissue interfaces (Jezzard, 1999; Eden and Zeffiro, 1997).

Additionally, random instability in the MR machine can result from imperfections in the B₁ field. The B₁ field is ideally a linear magnetic gradient that selects certain regions of tissue to be excited, thereby leading to the collection of single slices. Again, problems with this linear magnetic field can lead to blurring and geometric distortions in the data.

Noise from the Experimental Subject

As with other types of human studies, the experimental subjects can lead to large amounts of bias and variability in the data. While the subjects themselves have a great deal of intrinsic variability due to differences in brain sizes, shapes, and functionality in general, the subjects can also introduce additional variability that will “drown out” the desired results from brain activity if the investigator is not careful.

One important source of noise from the experimental subject is due to head motion. As previously described, BOLD fMRI studies compare very small regions of brain tissue across a sequence of images that are taken over the course of several minutes. While BOLD has the advantage that it requires no exogenous contrast agents, its measurable effects are very small. Typical changes in the MR signal due to BOLD are on the order of 1–5%, making this technique highly susceptible to noise. If the subject makes a small movement during the scan, adjacent voxels, which can vary in signal value by more than 10%, cause distortions in the recorded signal information and can lead to false negative and false positive regions of activation (Eddy and Young, 2000; Eddy et al., 1996b).

Thus, to obtain valid fMRI data, the subject must remain motionless throughout the scanning period. Motion has been shown to be correlated with stimulus related events during visual and motion stimulation, thereby contributing to the likelihood that the computed regions of activation are due to motion artifact rather than neural activity (Hajnal et al., 1994). The amount of subject motion has also been shown to increase over time during the course of a scanning session (Green et al., 1994). Additionally, children, elderly subjects, and subjects with mental disorders tend to move more than healthy young adults, thereby increasing the difficulty of studying these subjects using fMRI.

A second source of error from the experimental subject is due to “physiological noise”, which is noise that results from the subject’s heart beat and respiration. This type of complex noise is thought to interfere with the MR data through various mechanisms. For example, the pulsatile motions of the brain and cerebral spinal fluid (CSF) induced from pressure changes during both the cardiac and respiratory cycle lead to volume changes within the head which cause displacement of tissue; see Dagli et al. (1999). Large organ movements due to respiration are also thought to cause fluctuations in the magnetic field, and effects of the oscillating cardiac cycle on the BOLD signal response are unknown; see, e.g., Hu et al. (1995), Stenger et al. (1999), Dagli et al. (1999).

There are many sources of noise associated with the experimental subject. Thermal noise is caused by atomic vibration that occurs at any temperature above absolute zero. Susceptibility artifacts arise from local sharp changes in magnetic susceptibility; these occur at the boundaries of tissue types and are typically greatest at air/tissue boundaries. Chemical shift artifacts arise from small changes in the Larmor frequency caused by the local chemical environment. For example, hydrogen as a component of water has a resonant frequency at 3 Tesla that is about 200 Hz higher than hydrogen as a component of fat. Typical pulse sequences include a “fat saturation pulse” to eliminate this effect.

External Noise

Interference from outside sources can also lead to distortions and artifacts in the data. Examples of interference sources include mechanical vibrations from other equipment in the building or passing vehicles, and 60 (or 50) Hertz RF noise from other nearby electrical equipment. These sources are usually considered before installing the MR machines, and precautions are normally taken. For example, an isolated foundation will reduce the effect of external sources of vibration; copper shielding will reduce the effect of nearby sources of microwave radiation, and iron shielding will reduce the effect of nearby electrical equipment (and help contain the magnetic field itself).

Modeling and Analysis

4.4

An Ideal Model

4.4.1

From a simple statistical perspective, fMRI data can be considered as a large number of individual voxel time series. If these individual time series were independent, one could use any of the models from traditional time series analysis. For example, one could treat the brain as a linear system modulating the stimulus as its input. A model for this system would simply estimate the coefficients of the transfer function. Of course, there are other inputs such as heartbeat and respiration as well as other sources of data interference which would also need to be included. Thus, an ideal model would consist of an equation for each voxel that accounts for the true brain signal as well as all sources of systematic and random noise. By estimating all sources of noise and removing them appropriately for each voxel, an accurate estimate of brain activity could be found.

As can be imagined, an ideal model for fMRI data activation is highly impractical. A typical fMRI experiment may consist of $128 \times 128 \times 32$ voxels; therefore, a model that consists of a separate equation for each voxel would be quite cumbersome. Furthermore, the mere identification of each and every noise source in an fMRI experiment alone is a difficult task. Thereby a precise quantification of the effects of each noise source would be nearly impossible. In order to model and analyze fMRI data in a practical manner, researchers often take an approach

that is similar to that of solving any giant problem; that is, by breaking it down into piece-wise, practical steps. This approach allows for easier understanding of each step that is carried out (since they are performed one at a time) while also making the analysis computationally manageable. The piece-wise analysis steps involve first modeling and removing identifiable noise sources, then statistically evaluating the corrected data to estimate the amount of brain activation.

4.4.2 **A Practical Approach**

Preprocessing of the Data: Removal of Bias and Variance

Basically, two approaches can be employed to correct for known sources of bias and variance in fMRI data. These are correction of the data at the time of collection (preprocessing) and correction after data collection (post-processing). We use both approaches, often in tandem. For example, we use various forms of head restraint to (partially) eliminate head motion (preprocessing) and in addition we use post-processing to correct the data for head motion in addition.

We now give several examples of how data correction can be done in a post-processing manner. Some of these examples outline how many fMRI data processing steps are currently carried out using FIASCO (Functional Image Analysis Software – Computational Olio), which is a software collection that has been developed by the authors of this paper together with others and is currently used by numerous groups who analyze fMRI data. Details can be found at <http://www.stat.cmu.edu/~fiasco>.

These examples also demonstrate that large amounts of computation are often needed to remove bias and variance in fMRI data.

Noise from the equipment will first be addressed. Baseline noise or DC-shift noise can be corrected fairly easily, as it is a well understood source of noise in the data. To adjust for baseline noise, the idea that the k-space data should (approximately) be oscillating around 0 at the highest spatial frequencies is taken into account. If this is not the case in the true data, the mean value at which the high frequency data is oscillating is computed, and the k-space data is shifted by a constant prior to correct for this source of noise (see Eddy et al., 1996a).

Nyquist ghosts are also well understood and can therefore be corrected fairly easily. These ghosts arise from small mistimings in the gradients with respect to the data collection. To correct for these ghosts, a phase shift is applied to each line (the same shift for each line in the x -direction) of the complex-valued k-space data in order to move the data back into its correct location. The best phase shift for correction is estimated from the data by finding a value which minimizes the magnitude of the ghosts. Typically, the center portion of the top and bottom few lines is chosen as the target (see Eddy et al., 1996a).

Magnetic field inhomogeneities may or may not vary with time. Those which do not vary with time can be corrected by “shimming” the magnet. Small magnetic objects are placed around the magnet in such a way as to reduce the inhomogeneity. Inhomogeneities associated with the subject being scanned can be corrected by dynamic shimming of the field. This is a procedure performed by the technologist

at the time of the experiment. For further details see, e.g., Jezzard (1999) or Eden and Zeffiro (1997).

To reduce the effects of magnetic field distortions in a post-processing manner, the following procedure can be carried out. First a phase map can be computed from the image information in the fMRI data. This map is thought to give an estimate of regions of inhomogeneities in the data. Next, a 2-D polynomial can be fit to the field map, which is then converted into a pixel shift map. The shifted pixels are moved back to their proper locations in order to correct for the magnetic field inhomogeneities; further details may be found in Jezzard and Balaban (1995).

Noise from the experimental subject should also be corrected to improve data quality. As mentioned in the previous section, head motion is major problem in fMRI, since even relatively small amounts of head motion can lead to false positive and false negative regions of activation. Many techniques have been considered to reduce the amount of head motion in fMRI data. These (again) can be classified into the categories of preprocessing and post-processing.

To reduce the amount of head motion that occurs at the time of the fMRI scan, external restraining devices are often used. These devices range from pillows and straps to dental bite bars and even thermoplastic face masks; see Green et al. (1994). Head restraints can greatly reduce the amount of head motion but unfortunately cannot alleviate head motion (or at least brain motion) altogether; see Friston et al. (1996). In fact, Zeffiro (1996) have suggested that some types of restraints can paradoxically increase head motion because of the discomfort they cause at pressure points. Furthermore, some types of restraints may not be appropriate for certain subject types such as small children and mentally disturbed subjects.

Also experimental design can be used to reduce the amount of apparent head motion. Often head motion is associated with presentation of a stimulus or with physical response to the stimulus. Careful experimental design can eliminate or largely alleviate such effects.

A second way to reduce the effect of head motion at the time of the scan is through the use of navigator echos. Navigator echos are used before slice acquisition in order to detect displacements of the head. These displacements are then used to adjust the plane of excitation of the collected image accordingly. Examples include the use of navigator echos to correct for displacements in the z -direction (see Lee et al., 1996), and the use of navigator echos to correct for inter-image head rotation (see Lee et al., 1998).

A final example of a prospective method to reduce head motion is a visual feedback system that was developed by Thulborn (1999). This system provides a subject with information about their head location through a visual feedback system. By using this device the subject can tell if their head has moved during the scan and can immediately correct for it.

Preprocessing techniques have many benefits. For example, the collected fMRI data presumably has less head motion than it would have without the adaptation to reduce for head motion. Therefore, there is less need for post-processing of the data and less need to resample or interpolate the fMRI data; interpolation of fMRI data can introduce additional errors. On the other hand, these techniques often

require specialized collection sequences or hardware, which can further necessitate specialized analysis software. These techniques can also lead to the need for longer data collection times.

Post-processing techniques are often more feasible for fMRI researchers because they do not require specialized collection sequences or hardware. Post-processing is mainly carried out through image registration, which involves two main steps. The first is estimation of how much motion has occurred in the collected data, and the second is correction of the estimated motion through resampling (or interpolation). This process can be carried out in two dimensions for estimation and correction of in-plane motion or three dimensions for whole head motion.

To estimate how much head motion has occurred in an fMRI data set, a reference image must first be chosen. Any image can be chosen for this purpose (typically, the first or middle image in the series), but a composite image such as the mean can also be used. Next, each image in the series is compared to the reference image using a chosen feature of the images. Image intensity is often the chosen feature, but anatomical landmarks can also be used. Mathematically, the comparison process finds the geometrical shift required between the reference image and the target image to minimize an optimization criteria. The shift that minimizes this criteria is considered to be the amount of motion that has occurred.

For two dimensional rigid motion between the current image and the reference image we consider translations in x and y , and in-plane rotations of α . In three dimensions rigid motion is even more complicated because translations can occur in x , y , and z , and rotations can occur in the α , θ , or γ directions. Criteria to be minimized between the reference image and the current image have included weighted mean square error (see Eddy et al., 1996b), variance of the image ratios (Woods et al., 1992), and mutual information (Kim et al., 1999).

Once the motion estimates are computed, the data are moved back to their appropriate locations. This is carried out by resampling or interpolation. If data relocation is not performed in the best possible way, false correlations between image voxels can be introduced. Eddy et al. (1996b) developed a Fourier interpolation method to prevent introduction of these errors. Fourier interpolation is based on the Fourier shift theorem and uses the fact that a phase shift in k-space is equal to a voxel shift in image space. By this property, translations are corrected using phase shifts, and analogously rotations are corrected by k-space rotations. Rotations in k-space are implemented by factoring the two-dimensional rotation into a product of three shearing matrices. A two-dimensional rotation matrix

$$\begin{pmatrix} \cos \alpha & -\sin \alpha \\ \sin \alpha & \cos \alpha \end{pmatrix}$$

can be written as

$$\begin{pmatrix} 1 - \tan \frac{\alpha}{2} & \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ \sin \alpha & 1 \end{pmatrix} \begin{pmatrix} 1 - \tan \frac{\alpha}{2} & \\ 0 & 1 \end{pmatrix}$$

Once the rotations are represented by three shearing steps, these too can be corrected using phase shifts in Fourier space; for details see Eddy et al. (1996b). In a comparison of different types of fMRI data interpolation algorithms, Eddy and Young (2000) found that full Fourier interpolation was the only method which completely preserved the original data properties. Table 4.1 is taken from that paper and shows the error introduced by each method in performing the following motion. Each image was rotated by $\pi/64$ radians, translated 1/4 pixel on the diagonal, translated back 1/4 pixel, and rotated back $\pi/64$ radians. Many of the algorithms had major errors at the edges of the image so the statistics were only computed over a central portion where no edge effects occurred.

Although three dimensional head motion seems more realistic for correction of subject movements in fMRI research, the process of estimating the six motion parameters that jointly minimize the optimization criteria can be computationally expensive. Also note that these six parameters only account for rigid body head motion. Data is actually collected one slice at a time; each slice is collected at a slightly different time. Consequently, there is no three-dimensional image of the head which can be used as a target for three dimensional registration. In spite of these considerations, the use of post-processing motion correction techniques does not necessitate specialized collection equipment, and if done correctly, can be quite accurate for small motions of the head. Large head movements, greater than say a millimeter or two, usually affect the data so much that it cannot reasonably be used.

A second significant noise source from the experimental subject is physiological noise, which is mainly attributed to the subject's heart beat and respiration. Many methods have been introduced to reduce the noise and variability caused by physiological noise. These again have included techniques that address the problem in a preprocessing manner, and techniques that reduce this noise during post-

Table 4.1. Mean Square Difference between original brain image and "motion-corrected" image averaged over the central 40 by 40 pixel sub-image of the original 64 by 64 pixel image. The image was rotated $\pi/64$ radians, shifted 1/4 pixel diagonally and then translated back and rotated back to its original position; both motions were performed by the same algorithm so any differences from the original are due to the algorithm

Method	MSD
Fourier	0.00
WS16	742.86
WS8	1452.98
WS4	3136.68
NN	3830.08
Quintic	8906.20
Cubic	13,864.46
WS2	28,455.73
Linear	28,949.22

processing of the data by various modeling and subtraction techniques (Biswal et al., 1996; Chuang and Chen, 2001; Hu et al., 1995; Glover et al., 2000).

Acquisition gating is most commonly used for the collection of cardiac MRI data, but it can also be used for fMRI. This preprocessing technique only collects the MRI data at certain points of the cardiac cycle so that noise effects due to cycle variations can be reduced (Guimaraes et al., 1996). Although useful for cardiac MRI, it is not as practical for fMRI research because it does not allow for continuous and rapid data collection, which is usually desirable for fMRI research.

Post-processing techniques to correct for physiological noise have included retrospective image sorting according to the phase of the cardiac cycle (Dagli et al., 1999), phase correction through the assumption that the phase is uniform for each point in k-space over many images (Wowk et al., 1997), and digital filtering (Biswal et al., 1996). Each of these techniques have certain benefits in physiological

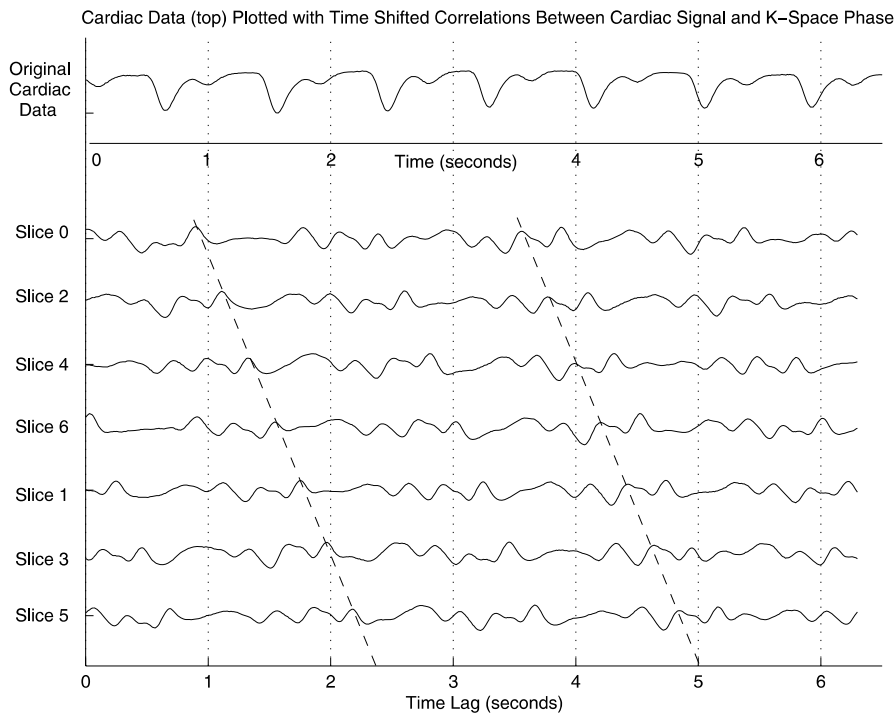


Figure 4.6. Cardiac data shown in relation to the correlation coefficients computed between k-space phase and cardiac data as a function of time lag for a spatial frequency point in k-space. To demonstrate that the time lag is slice dependent, slices are shown in the order of collection rather than in anatomical order. Each tick-mark on the y-axis represents a unit value of 0.5. It can be clearly noted that the correlations between the phase and cardiac data are cyclic with a period equal to the TR (1.5 seconds in this case) of the study. Thus the *parallel diagonal lines* in the plot have a slope of $(1.5/7 = 0.214 \text{ seconds/slice})$.

noise correction; however, each can also compromise the fMRI data. For example, image reordering results in loss of temporal resolution, and digital filtering requires the ability to acquire images rapidly as compared to the physiological noise signals.

Other approaches to physiological noise correction have included modeling of the physiological data signals in the fMRI data and subtracting out their effects. For example, Hu modeled respiration with a truncated Fourier Series and subtracted this curve from the magnitude and phase components of the k-space data (Hu et al., 1995). Glover et al. (2000) carried out a similar Fourier modeling and subtraction procedure in image space. Alternatively, Mitra and Pesaran modeled cardiac and respiratory effects as slow amplitude, frequency-modulated sinusoids and have removed these components from an fMRI principal component time series, which was obtained using a spatial frequency singular value decomposition Mitra and Pesaran (1999).

In a more recent study carried out by the authors, cardiac data and respiratory data were found to be significantly correlated with the k-space fMRI data. These correlations are a function of both time and spatial frequency (McNamee and Eddy, 2003). Because of these correlations, the physiological effects can simply be reduced by first collecting the physiological data along with the fMRI data, regressing the fMRI data onto the physiological data (with the appropriate temporal considerations), and subtracting out the fitted effects (McNamee and Eddy, 2004). Figure 4.6 shows the cross-correlations between the k-space phase at one spatial frequency point and the cardiac data for one experiment.

In addition to direct modeling of known sources of noise, we also attempt to reduce the effect of unknown sources of noise. For example, we routinely replace outliers with less extreme values. As another example we currently remove a linear temporal trend from each voxel times series, although we have no explanation for the source of this linear trend.

Modeling and Analysis of the Data

The very first fMRI experiments claimed to have discovered brain activation based on simply taking the difference between the average image in one experimental condition from the average in an alternate experimental condition. There was no notion of variability. It quickly became apparent that accounting for the variability was very important and fMRI analyses started using t-tests, independently on each voxel. It then became clear that multiple testing was a very serious problem; we will discuss this in more detail below. The t-test (and for experiments with more than two levels, F-test) became the standard analysis method. However, it is clear that although the stimulus switches instantaneously between experimental conditions, the brain response (presumably being continuous) must transition to the new state and there will be some time lag before the transition is complete. This has led to a variety of ad hoc “models” for the shape and lag of this transition (Gamma functions, Poissons, etc.) (see, for example, Lange and Zeger, 1997).

There has been some work developing reasonable non-parametric models for the voxel time courses (Genovese, 2000) and there have been a number of “time

series” modeling approaches, using spectral analysis (Lange and Zeger, 1997) and AR models (Harrison et al., 2003).

Because a typical brain image contains more than 100,000 voxels, it is clear that choosing a significance level of 0.05 will, even under the null hypothesis of no difference in brain activity, lead to 5000 or more voxels being declared active. The earliest attempt to deal with this problem in fMRI was the split t-test, wherein the data were divided into two (or more) temporal subsets. T-tests were calculated separately within each subset and a voxel was declared active only if it was active in all subsets (Schneider et al., 1993). This certainly reduced the number of false positives, but obviously caused considerable loss of power.

Researchers quickly converged on the Bonferonni correction where one simply divides the significance level by the number of tests as a safe way of dealing with the problem. The loss of power is huge and some researchers started developing methods to compensate. Forman et al. (1995) proposed the contiguity threshold method, which relies on the presumption that if one voxel is active then adjacent voxels are likely to be active.

The false discovery rate (FDR) controlling procedure was then introduced as an alternate method for thresholding in the presence of multiple comparison testing. The FDR procedure allows the user to select the maximum tolerable FDR, and the procedure guarantees that *on average* the FDR will be no larger than this value. Details of the application of FDR to fMRI data can be found in Genovese et al. (2002).

As researchers in fMRI have discovered various general classes of statistical models, they have been applied to these data sets. They have had varying degrees of success and it is still the case that a linear model, often with both fixed and random effects, is the model of choice. It is reasonably well-understood and is not too far from what is actually believed. One critical problem is that as experimental designs have become more complex analyses have too; often researchers are fitting models blindly without fully understanding some of the underlying statistical issues. The more cautious return to their linear models.

4.5 Computational Issues

As mentioned previously, a typical fMRI data set might be about 1 GB in size and would take less than 15 minutes to collect. Four or more such data sets may be collected in a single experimental session with a single subject. An entire experiment might include 20 or more subjects, and each subject might be tested twice or more. Thus there would be 160 1 GB datasets to be analyzed. (We are currently running an experiment with 300 subjects and each will be tested at least twice; one has already been tested four times.)

With such large amounts of data, storage and analysis becomes an important issue. Standard analysis of the data can take many hours, and an organized storage system is recommended. Our recent experience is that we can process the data

at the rate of 2–3 MB per minute which implies that the entire experiment just described would require on the order of 1000 hours of processing time.

Several packages are available for analysis of fMRI data. A few of these will be discussed in the following section. An important point to mention before discussing these packages is that users of fMRI software should spend some time getting to know and understand the package they are using before carrying out their data processing. Questions that are important to consider, for example, may be the following. Are bias and variance in the data corrected as a routine part of data processing? If motion correction is carried out, how is this implemented? Is any additional error being introduced into the data as a result of this routine processing? What kind of modeling and comparisons of the data are being carried out, and what sort of thresholding is applied? We feel that these issues should be understood before drawing conclusions about the fMRI data, as variations in the processing and analysis may lead to variations in the results of the study.

Software Packages for fMRI Data Analysis

4.5.1

The large degree of complexity of fMRI data necessitates the use of pre-packaged software tools (unless, of course, one is an extremely ambitious programmer). Several packaged tools are described briefly below. Each software package has its benefits and limitations; we do not provide a detailed comparison here. We will focus a bit more on FIASCO, as we are part of the team that has developed and maintained this package; it is in wide use and we describe several of its unique tools.

One such software program for fMRI data processing is AFNI (Analysis of Functional NeuroImages). This software was developed by Robert Cox, formerly of the Medical College of Wisconsin, now at the National Institutes of Health; it is a free to all users. The software consists of a set of C programs for processing, analyzing, and displaying fMRI data. Currently, AFNI will run on most operating systems excluding Windows-based platforms. The program is interactive, and several of its benefits include its ability to read and switch between several different data formats, 3D viewing of data, the ability to transform data into Talairach coordinates, interactive thresholding of functional overlays onto structural images, and finally, a new feature entitled SUMA (Surface Mapping with AFNI) that adds the ability to perform cortical surface based functional imaging analysis using AFNI. The homepage for AFNI is located at <http://afni.nimh.nih.gov/afni/>.

Brain Voyager is a commercially available tool developed by Brain Innovation that can analyze and visualize both functional and structural MRI datasets. Brain Voyager can run on all major computer platforms including all current versions of Windows, as well as Linux/Unix and Macintosh systems. A user-interface is provided, and the program boasts several up-to-date features such as the ability to perform thresholding using the FDR technique, the ability to perform automatic brain segmentation, brain surface reconstruction, and cortex inflation and

flattening, and the ability to analyze and integrate diffusion tensor imaging data with other data types. More information about this product can be found at <http://www.brainvoyager.com/>.

The Statistical Parametric Mapping (SPM) software package is a suite of programs, originally developed by Karl Friston, to analyze SPECT/PET and fMRI data using a voxel-based approach. The SPM program is also free but requires Matlab, a licensed Mathworks product, to run. Typical analysis steps performed by SPM include spatial normalization and smoothing of the images, parametric statistical modeling at each voxel through the use of a general linear model, and assessment of the computed statistical images. New functionality of SPM (released in SPM2) can take into account more complex issues in fMRI research such as non-sphericity, which does not restrict the user to the assumption of identically and independently distributed errors in the selected data models. For more details about this and other SPM specifics, the reader should refer to <http://www.fil.ion.ucl.ac.uk/spm/>.

The VoxBo package advertises itself as “the software behind the brains”. The VoxBo software is free and was developed at the University of Pennsylvania through funding from NIDA and NIMH via a Human Brain Project/Neuroinformatics grant. A Unix-based platform is currently required to run VoxBo, and a unique feature of this software includes a job scheduling system for organizational purposes. Benefits of VoxBo also include the ability carry out several data pre-processing steps such as three dimensional motion correction as well as the ability to manipulate and model the data in its time-series form. A web page is available at <http://www.voxbo.org/>.

A group including the current authors developed an fMRI analysis package named FIASCO. This is a collection of software tools written primarily in C, designed to analyze fMRI data using a series of processing steps. Originally, FIASCO's main purpose was to read the raw fMRI data, process it in a series of steps to reduce sources of systematic error in the data, carry out statistical analysis, and create final brain maps showing regions of neural activation. While users still run their fMRI data through the standard FIASCO pipeline, FIASCO has expanded into a complex set of software tools in order to accommodate many other issues and problems that have come up during the past decade as the field of fMRI has grown.

One unique feature of FIASCO is that several of the data-processing steps for the purpose of reducing bias and noise are carried out in k-space. As mentioned previously, k-space is essentially a frequency space and is the domain in which the raw data is collected. Carrying out data correction in k-space can be beneficial for many reasons. For example, certain types of noise in the data (such as Nyquist ghosts, phase drifts, and physiological noise) can be more accurately modeled and removed in k-space than in image space (McNamee and Eddy, 2004). Also, images can be resampled in k-space without the need for interpolation or smoothing, both of which can introduce additional problems into the images (Eddy and Young, 2000).

If users elect to use the typical FIASCO pipeline for fMRI data analysis, tools that implement k-space correction are first carried out prior to performing the Fourier Transform. Both EPI and spiral data can be analyzed, and since both have different types of noise, unique pipelines are used for each. For example, typical

FIASCO steps for processing EPI data include baseline correction (aka. correction of DC shift), removal of Nyquist ghosts, motion correction, physiological noise correction, removal of outliers and removal of unexplained data trends. All of these correction steps with the exception of the last two are carried out in k-space. The final steps are to perform statistical analysis and create brain maps showing the activated areas. Users can elect to skip any of the steps of the FIASCO process. And, they have the opportunity to add their own unique steps to the process.

As the field of fMRI grows, so does the desire to be able to perform more complex analysis procedures on the data. The FIASCO programmers have accommodated these needs with the creation of several unique general purpose tools. For example, certain tools allow the user to easily manipulate fMRI data sets by cutting, pasting, permuting, or sorting the data. More complex general purpose tools include an rpn-math feature with built-in functions; this is essentially a calculator which allows the user to perform arithmetic calculations using Reverse Polish Notation looping over all the voxels in a data set. Another tool allows users to perform matrix multiplication on fMRI data sets, and a third tool can compute eigenvalues and eigenvectors of real symmetric matrices.

In addition to general purpose tools, FIASCO also has many special purpose tools that can perform very specific tasks in relation to the fMRI data. Some of these include tools that can convert between different data formats, tools that perform specific noise reduction steps, and tools that can compute summary statistics or perform different types of hypothesis tests on the data. A complete list of FIASCO's tools can be found on the web page at <http://www.stat.cmu.edu/~fiasco>.

The features and tools of FIASCO have allowed its users to manipulate and experiment with fMRI data sets in unique and interesting ways. Recently, FIASCO has been applied to many other kinds of data: genetic microarrays, protein gels, video, PET, CT, etc.

Other Computational Issues

4.5.2

Aside from fMRI analysis software, other types of useful computational packages in fMRI may include software that allows the researcher to easily design and implement a desired experimental tasks for subjects to carry out while in the MRI scanner. For example, an fMRI study may focus on activation patterns during a short-term memory task. Thus an experiment engaging short-term memory would need to be carefully designed and projected into the small space of the MRI machine. The scientist would also need some sort of feedback from the subject to ensure that the task was being performed properly.

Conclusions

4.6

Functional MRI is a new and exciting method for studying the human brain as it functions in its living, natural state. As the field grows and the methodology

improves, so do the many computational and statistical problems that accompany the storage, processing, analysis, and interpretation of the data. In this chapter we have briefly summarized some of the complexities and limitations of the method as well as describing some of the approaches available for addressing these complexities and limitations. The field of fMRI will, no doubt, continue to broaden and expand. As it does so, the continued integration of scientists and researcher from many disciplines will be necessary for fMRI to reach its full potential and to help to uncover one of the greatest mysteries of humankind.

References

- Belliveau, J.W., Kennedy, D.N., McKinstry, R.C., Buchbinder, B.R., Weisskoff, R.M., Cohen, M.S., Vevea, J.M., Brady, T.J. and Rosen, B.R. (1991). Functional mapping of the human visual cortex by magnetic resonance imaging. *Science*, 254: 716–719.
- Biswal, B., DeYoe, E.A. and Hyde, J.S. (1996). Reduction of physiological fluctuations in fMRI using digital filters. *Magn. Reson. Med.*, 35: 107–113.
- Buxton, R.B. (2002). *Introduction to Functional Magnetic Resonance Imaging: Principles and Techniques*, Cambridge University Press, Cambridge.
- Cherry, S.R. and Phelps, M.E. (1996). Imaging brain function with positron emission tomography. In Toga, A.W. and Mazziotta, J.C. (eds), *Brain Mapping: The Methods*. Academic Press, New York.
- Chuang, K.H. and Chen, J.H. (2001). IMPACT: Image-based physiological artifacts estimation and correction technique for functional MRI. *Magn. Reson. Med.*, 46: 344–353.
- Dagli, M.S., Ingeholm, J.E. and Haxby, J.V. (1999). Localization of cardiac-induced signal change in fMRI. *NeuroImage*, 9: 407–415
- Eddy, W.F., Fitzgerald, M., Genovese, C.R., Mockus, A. and Noll, D.C. (1996a). Functional imaging analysis software – computational olio. In Prat, A. (ed), *Proceedings in Computational Statistics*, Physica-Verlag, Heidelberg.
- Eddy, W.F., Fitzgerald, M. and Noll, D.C. (1996b) Improved image registration by using Fourier interpolation. *Magn. Reson. Med.*, 36: 923–931.
- Eddy, W.F., Fitzgerald, M., Genovese, C., Lazar, N., Mockus, A. and Welling, J. (1999). The challenge of functional magnetic resonance imaging. *J. Comp. and Graph. Stat.*, 8(3), 545–558. Adapted from: The challenge of functional magnetic resonance imaging. In *Massive Data Sets, Proceedings of a Workshop*, National Academy Press, Washington, D.C., pp. 39–45, 1996.
- Eddy, W.F. and Young, T.K. (2000). Optimizing the resampling of registered images. In Bankman, I.N. (ed), *Handbook of Medical Imaging: Processing and Analysis*. Academic Press, New York.
- Eden, G.F. and Zeffiro, T.A. (1997). PET and fMRI in the detection of task-related brain activity: Implications for the study of brain development. In Thacher, R.W., Lyon, G.R., Rumsey, J. and Krasnegor, N.K. (eds), *Developmental NeuroImaging: Mapping the Development of Brain and Behavior*, Academic Press, San Diego.

- Finger, S. (1994). *Origins of Neuroscience: A History of Explorations Into Brain Function*, Oxford University Press, Oxford.
- Forman, S.D., Cohen, J.D., Fitzgerald, M., Eddy, W.F., Mintun, M.A. and Noll, D.C. (1995). Improved assessment of significant change in functional magnetic resonance imaging (fMRI): Use of a cluster size threshold. *Magn. Reson. Med.*, 33: 636–647.
- Friston, K.J., Williams, S., Howard, R., Frackowiak, R.S.J. and Turner, R. (1996). Movement-related effects in fMRI time-series. *Magn. Reson. Med.*, 35: 346–355.
- Gaillard, W.D., Grandin, C.B. and Xu, B. (2001). Developmental aspects of pediatric fMRI: Considerations for image acquisition, analysis, and interpretation. *NeuroImage*, 13, 239–249.
- Genovese, C.R. (2000). A Bayesian time-course model for functional magnetic resonance imaging (with discussion). *J. Amer. Statist. Assoc.*, 95: 691–703.
- Genovese, C.R., Lazar, N.A. and Nichols, T.E. (2002). Thresholding of statistical maps in neuroimaging using the false discovery rate. *NeuroImage*, 15: 870–878.
- Glad, I. and Sebastiani, G. (1995). A Bayesian approach to synthetic magnetic resonance imaging. *Biometrika*, 82: 237–250.
- Glover, G.H., Li, T.Q. and Ress, D. (2000). Image-based method for retrospective correction of physiological motion effects in fMRI: RETROICOR. *Magn. Reson. Med.*, 44: 162–167.
- Green, M.V., Seidel, J., Stein, S.D., Tedder, T.E., Kempner, K.M., Kertzman, C. and Zeffiro, T.A. (1994). Head movement in normal subjects during simulated PET brain imaging with and without head restraint. *J. Nucl. Med.*, 35: 9, 1538–1546.
- Guimaraes, A.R., Melcher, J.R., Talavage, T.M., Baker, A.J., Rosen, B.R. and Weisskoff, R.M. (1996). Detection of inferior colliculus activity during auditory stimulation using cardiac-gated functional MRI with T_1 correction. In *NeuroImage*, 2nd International Conference on Functional Mapping of the Human Brain.
- Hajnal, J.V., Myers, R., Oatridge, A., Schwieso, J.E., Young, I.R. and Bydder, G.M. (1994). Artifacts due to stimulus correlated motion in functional imaging of the brain. *Magn. Reson. Med.* 31: 283–291.
- Harrison, L., Penny, W.D. and Friston, K. (2003). Multivariate Autoregressive Modeling of fMRI time series. *NeuroImage*, 19(4): 1477–1491.
- Hu, X., Le, T.H., Parrish, R. and Erhard, P. (1995). Retrospective estimation and correction of physiological fluctuation in functional MRI. *Magn. Reson. Med.*, 34: 201–212.
- Jezzard, P. (1999). Sources of distortion in functional MRI data. *Hum. Brain Map.*, 8: 80–85.
- Jezzard, P. and Balaban, R.S. (1995). Correction for geometric distortions in echo planar images from B_0 field variations. *Magn. Reson. Med.*, 34: 65–73.
- Kim, B., Boes, J.L., Bland, P.H., Chenevert, T.L. and Meyer, C.R. (1999). Motion correction in fMRI via registration of individual slices into an anatomical volume. *Magn. Reson. Med.*, 41: 964–972.

- Kwong, K.K., Belliveau, J.W., Chesler, D.A., Goldberg, I.E., Weisskoff, R.M., Poncelet, B.P., Kennedy, D.N., Hoppel, B.E., Cohen, M.S., Turner, R., Cheng, H., Brady, T.J. and Rosen, B.R. (1992). Dynamic magnetic resonance imaging of human brain activity during primary sensory stimulation. *Proc. Natl. Acad. Sci. U.S.A.*, 89: 5675.
- Lange, N. (1996). Statistical approaches to human brain mapping by functional magnetic resonance imaging. *Statist. Med.*, 15: 389–428.
- Lange, N. and Zeger, S.L. (1997). Non-linear Fourier time series analysis for human brain mapping by functional magnetic resonance imaging (with discussion). *Appl. Stat.*, 46: 1–29.
- Lazar, N.A., Genovese, C.R., Eddy, W.F. and Welling, J. (2001). Statistical issues in fMRI for brain imaging. *Int. Stat. Rev.*, 69: 105–127.
- Lee, C.C., Jack, C.R., Grimm, R.C., Rossman, P.J., Felmlee, J.P., Ehman, R.L. and Riederer, S.J. (1996). Real-time adaptive motion correction in functional MRI. *Magn. Reson. Med.*, 36: 436–444.
- Lee, C.C., Grimm, R.C., Manduca, A., Felmlee, J.P., Ehman, R.L., Riederer, S.J. and Jack, C.R. (1998). A prospective approach to correct for inter-image head rotation in FMRI. *Magn. Reson. Med.*, 39: 234–243.
- Logothetis, N.K. (2002). The neural basis of the blood-oxygen level dependent functional magnetic resonance imaging signal. *Philos. Trans. R. Soc. Lond. B. Biol. Sci.*, 357(1424): 1003–1037.
- McAllister, T.W., Saykin, A.J., Flashman, L.A., Sparling, M.B., Johnson, S.C., Guerin, S.J., Mamourian, A.C., Weaver, J.B. and Yanofsky, N. (1999). Brain activation during working memory 1 month after mild traumatic brain injury: A functional MRI study. *Neurol.*, 53: 1300–1308.
- McNamee, R.L. and Eddy, W.F. (2003). Correlations between cardiac data and fMRI data as a function of spatial frequency and time. In *Proceedings of the 2003 25th Annual International Conference of the IEEE-EMBS Society*. Cancun, Mexico.
- McNamee, R.L. and Eddy, W.F. (2004). Examination and removal of the spatial and time-related effects of physiological noise in fMRI data. (in preparation)
- Mitra, P.P. and Pesaran, B. (1999). Analysis of dynamic brain imaging data. *Biophys. J.*, 78: 691–708.
- Ogawa, S., Lee T.-M., Nayak, A.S. and Glynn, P. (1990). Oxygenation-sensitive contrast in magnetic resonance image of rodent brain at high magnetic fields. *Magn. Reson. Med.*, 14: 680–78.
- Ogawa, S., Tank, D.W., Menon, D.W., Ellermann, J.M., Kim, S., Merkle, H. and Ugurbil, K. (1992). Intrinsic signal changes accompanying sensory stimulation: Functional brain mapping using MRI. *Proc. Natl. Acad. Sci. U.S.A.*, 89: 5951–5955.
- Poldrack, R.A. (2000). Imaging brain plasticity: Conceptual and methodological issues – A theoretical review. *NeuroImage*, 12: 1–13.
- Rosen, A.C., Prull, M.W., O'Hara, R., Race, E.A., Desmond, J.E., Glover, G.H., Yesavage, J.A. and Gabrieli, J.D. (2002). Variable effects of aging on frontal lobe contributions to memory. *NeuroReport*, 3(8): 2425–2428.

- Schneider, W., Noll, D.C. and Cohen, J.D. (1993). Functional topographic mapping of the cortical ribbon in human vision with conventional MRI scanners. *Nature*, 365: 150–153.
- Stenger, V.A., Peltier, S., Boada, F.E. and Noll, D.C. (1999). 3D spiral cardiac/respiratory ordered fMRI data acquisition at 3 Tesla. *Magn. Reson. Med.*, 41: 983–991.
- Thulborn, K.R., Waterton, J.C., Matthews, P.M. and Radda, G.K. (1982). Oxygenation dependence of the transverse relaxation time of water protons in whole blood at high field. *Biochem. Biophys. Acta.*, 714: 265–270.
- Thulborn, K.R. (1999). Visual feedback to stabilize head position for fMRI. *Magn. Reson. Med.*, 41: 1039–1043.
- Turner, R. (1995). Functional mapping of the human brain with magnetic resonance imaging. *Sem. in the Neurosci.*, 7: 179–194.
- Villringer, A. (2000). Physiological Changes During Brain Activation. In Moonen, C.T.W. and Bandettini, P.A. (eds), *Functional MRI*, Springer-Verlag, Berlin.
- Weisskoff, R.M. (1996). Simple measurement of scanner stability for functional NMR imaging of activation in the brain. *Magnetic Resonance in Medicine*, 36: 643–645.
- Weisskoff, R.M. and Kiihne, S. (1992). MRI susceptometry: Image-based measurement of absolute susceptibility of MR contrast agents and human blood. *Magn. Reson. Med.*, 24: 375–83.
- Woods, R., Cherry, S. and Mazziotta, J. (1992). Rapid automated algorithm for aligning and reslicing PET images. *J. Comp. Ass. Tomog.*, 16: 620–633.
- Wowk, B., McIntyre, M.C. and Saunders, J.K. (1997). k-space detection and correction of physiological artifacts in fMRI. *Magn. Reson. Med.*, 38: 1029–1034
- Zeffiro, T. (1996). Clinical functional image analysis: Artifact detection and reduction. *NeuroImage*, 4: S95–S100.

Network Intrusion Detection

IV.5

David J. Marchette

5.1	<i>Introduction</i>	1030
5.2	<i>Basic TCP/IP</i>	1030
5.3	<i>Passive Sensing of Denial of Service Attacks</i>	1032
5.4	<i>Streaming Data</i>	1033
5.5	<i>Visualization</i>	1037
5.6	<i>Profiling and Anomaly Detection</i>	1042
5.7	<i>Discussion</i>	1048

5.1**Introduction**

Attacks against computers and the Internet are in the news every week. These primarily take the form of malicious code such as viruses and worms, or denial of service attacks. Less commonly reported are attacks which gain access to computers, either for the purpose of producing damage (such as defacing web sites or deleting data) or for the opportunities such access provides to the attacker, such as access to bank accounts or control systems of power stations. This chapter will discuss some of the areas in which computational statistics can be applied to these and related problems.

Several books are available that describe the basic ideas in intrusion detection. These include (Amoroso, 1999, anonymous, 1997, Bace, 2000, Escamilla, 1998, Marchette, 2001, Northcutt et al., 2001, Proctor, 2001). Intrusion detection is typically split into two separate problems. Network intrusion detection typically looks at traffic on the network, while host based intrusion detection involves collecting data on a single host. Both involve very large and complex data sets, and both have aspects that lend themselves to statistical solutions. We will only touch on a few such; the reader is encouraged to investigate the references.

There are two basic approaches to network intrusion detection. Most existing systems rely on signatures of attacks. This approach relies on some set of features that can be extracted from the data that indicate the existence of an attack. This is analogous to the virus scanners, which look for a sequence of bytes that are indicative of a virus. In the network realm, this could be attempts to access services that are denied, malformed packets, too many failed attempts to log in, et cetera. The second approach is anomaly detection. The “normal” activity of the network is modeled, and outliers are indicative of attacks. The definition of “normal” is dependent on the type of attacks that one is interested in, and requires statistical models.

This chapter will first describe the basics of the TCP/IP protocol, sufficient to understand the data and the examples given. Then we will look at detecting denial of service attacks, and estimating the number of attacks on the Internet. Network data is streaming data, and we will discuss this and some areas in which computational statistics can play a part. This will lead to a discussion of simple visualization techniques applied to network data, with some discussion of the types of insights that can be gained from this. We will then take a detour from network data and consider profiling. This will illustrate a type of anomaly detection, which will then be discussed within a network context.

5.2**Basic TCP/IP**

When you visit a web site, your request and the response data are sent as a series of packets, each consisting of a header containing addressing and sequencing information, and a payload or data section in which the information resides.

Packets are typically relatively small (less than 1500 bytes). In order to analyze the traffic and detect attacks, one needs to collect the packets, and may need to process either the header or the payload. We will (somewhat arbitrarily) denote an attack that can be detected by investigating the header only a “network attack” while leaving those that require investigation of the payload in the “host attack” realm.

One reason for this distinction is encryption. If the data are encrypted (for example, data from a secure web site), the header remains in the clear, and so this information is still available for analysis by the statistician. The payload is inaccessible (assuming a sufficiently strong encryption scheme) and so cannot be used to detect attacks until it is decrypted at the destination host. For this reason (and others), we consider any attack that requires investigation of the data in a packet to be better detected at the host than on the network.

There are several protocols used on the Internet to ensure a level of performance or reliability in the communication. We will briefly discuss TCP (the Transmission Control Protocol), since it is one of the most important ones, and will allow us to discuss a class of denial of service attacks. For more information about the various protocols, see (Stevens, 1994).

First, however, it is necessary that we discuss the Internet Protocol (IP). This protocol is not reliable, in the sense that there is no mechanism in place to ensure that packets are received. The IP header contains the source and destination IP addresses, which are 32-bit integers identifying the sending and receiving computer for the packet. There are other fields in the packet that are used to control the routing of the packet, et cetera, but we will not dwell on these here. As always, interested readers should investigate (Stevens, 1994) or any of the many books on the TCP/IP protocol suite.

Since IP is unreliable, a packet sent may or may not reach its destination, and if it does not, there is no guarantee that anyone will notice. Thus, a more reliable protocol is required. TCP implements a reliable two way communication channel, and is used for web, email, and many other user applications. The TCP header is shown in Fig. 5.1. The important fields, for this discussion, are the ports, sequence numbers and flags.

The ports are a method for identifying a specific session, and can be thought of as a 16-bit addition to the IP address that uniquely determines the session. Ports are also used to identify the application requested. For example, port 80 is the standard web port, and web browsers know that in order to obtain a web page from a server they need to make a connection on this port.

To initiate and maintain a connection, the flags and sequence numbers are used. The TCP protocol requires a three-way handshake to initiate a connection. First the client sends a SYN packet (in this manner we will denote a packet with only the SYN flag set; similarly with other flag combinations) to the server. The server responds with a SYN/ACK packet, acknowledging the connection. The client then finalizes the connection with an ACK packet. Sequence numbers are also passed, and tracked to ensure that all sent packets are received and acknowledged, and to allow the reconstruction of the session in the correct order. Packets that are not acknowledged are resent, to ensure that they are ultimately received and processed.

Source Port		Destination Port	
Sequence Number			
Acknowledgment Number			
Length	Reserved	Flags	Window Size
Checksum		Urgent Pointer	
Options (if any)			

Figure 5.1. The TCP header. The header is to be read left to right, top to bottom. A row corresponds to 32 bits

Once a session has been instantiated through the three-way handshake, packets are acknowledged with packets in which the ACK flag is set. In this manner the protocol can determine which packets have been received and which need to be resent. If a packet has not been acknowledged within a given time, the packet is resent, and this can happen several times before the system determines that something has gone wrong and the session is dropped (usually by sending a reset (RST) packet). Note that this means that if there is no response to the SYN/ACK packet acknowledging the initiation of the session there will be a period (of several seconds) in which the session is kept open by the destination host as it tries resending the SYN/ACK hoping for a response. This is the basis of some denial of service attacks, which we will discuss in the next section.

Passive Sensing of Denial of Service Attacks

5.3

The TCP protocol provides a simple (and popular) method for denial of service attacks. The server has a finite number of connections that it can handle at a time, and will refuse connections when its table is full. Thus, if an attacker can fill the table with bogus connections, legitimate users will be locked out.

This attack relies on two fundamental flaws in the protocols. The first is that the source IP address is never checked, and thus can be “spoofed” by putting an arbitrary 32 bit number in its place. Second, the three-way handshake requires the third (acknowledgment) packet, and the server will wait several seconds before timing out a connection. With each requested connection, the server allocates a space in its table and waits for the final acknowledgment (or for the connection to time out). The attacker can easily fill the table and keep it filled by sending spoofed SYN packets to the server.

Thus, the attacker sends many SYN packets to the server, spoofed to appear to come from a large number of different hosts. The server responds with SYN/ACK

packets to these hosts, and puts the connection in its table to await the final ACK, or a time-out (usually several seconds). Since the ACK packets are not forthcoming, the table quickly fills up, and stays full for as long as the attacker continues to send packets.

There are clever ways to mitigate this problem, which can keep the table from filling up. One, the “SYN-cookie” involves encoding the sequence number of the SYN/ACK in a way that allows the server to recognize legitimate ACK packets without needing to save a spot in the table for the connection. However, even these can be defeated through a sufficiently high volume attack.

These unsolicited SYN/ACK packets can be observed by any network sensor, and thus provide a method for estimating the number and severity of such attacks throughout the Internet. These unsolicited packets are referred to as *backscatter*. They may take other forms than SYN/ACK packets, depending on the type of packet sent in the attack. See (Moore et al., 2001, Marchette, 2002, Marchette,) for more information.

Typically, the attacker first compromises a large number of computers, using special distributed attack software, and it is these computers that launch the attack. This makes it very difficult to block the attack, and essentially impossible to track down the attacker, at least through information available to the victim.

Backscatter packets provide several opportunities for statistical analysis. They allow the estimation of the number of attacks on the Internet in real time. One may be able to estimate the severity of the attacks and number of attackers. Finally, it may be possible to characterize different types of attacks or different attack tools and identify them from the pattern of the packets. Some initial work describing some of these ideas is found in (Giles et al., 2003).

A network sensor is a computer that captures packets (usually just the packet headers) as they traverse the network. These are usually placed either just before or just after a firewall to collect all the packets coming into a network. Through such a system, one can observe all the unsolicited SYN/ACK packets addressed to one of the IP addresses owned by the network.

Note that this means that only a fraction of the backscatter packets resulting from the attack are seen by any sensor. If we assume that the sensor is monitoring a class B network (an address space of 65,536 IP addresses), then we observe a random sample of $1/65,536$ of the packets, assuming the attack selects randomly from all 2^{32} possible IP addresses. This points to several areas of interest to statisticians: we observe a subset of the packets sent to a subset of the victims, and wish to estimate the number of victims, the number of packets sent to any given victim, and the number of attackers for any given victim.

Streaming Data

Network packets are streaming data. Standard statistical and data mining methods deal with a fixed data set. There is a concept of the size of the data set (usually

denoted n) and algorithms are chosen based in part on their performance as a function of n . In streaming data there is no n : the data are continually captured and must be processed as they arrive. While one may collect a set of data to use to develop algorithms, the nonstationarity of the data requires methods that can handle the streaming data directly, and update their models on the fly.

Consider the problem of estimating the average amount of data transferred in a session for a web server. This is not stationary: there are diurnal effects; there may be seasonal effects (for example at a university); there may be changes in the content at the server. We'd like a number calculated on a window of time that allows us to track (and account for) the normal trends and detect changes from this normal activity.

This requires some type of windowing or recursive technique. The recursive version of the sample mean is well known:

$$\bar{X}_n = \frac{n-1}{n}\bar{X}_{n-1} + \frac{1}{n}X_n.$$

Replacing n on the right hand side with a fixed constant N implements an exponential window on the mean. This was exploited in the NIDES intrusion detection system (Anderson et al., 1995). Similar techniques can be used to compute other moments. An alternative formulation is:

$$\hat{X}_n = (1 - \theta)X_{n+1} + \theta\hat{x}_{n-1},$$

for $0 < \theta < 1$. θ may be fixed or may itself change based on some statistic of the data.

In fact, the kernel density estimator has a simple recursive version, that allows the recursive estimate of the kernel density estimator at a fixed grid of points. (Yamato, 1971, Wegman and Davies, 1979) give two versions of this:

$$\hat{f}_n(x) = \frac{n-1}{n}\hat{f}_{n-1}(x) + \frac{1}{nh_n}K\left(\frac{x-X_n}{h_n}\right)$$

$$\check{f}_n(x) = \frac{n-1}{n}\left(\frac{h_{n-1}}{h_n}\right)^{1/2}\check{f}_{n-1}(x) + \frac{1}{nh_n}K\left(\frac{x-X_n}{h_n}\right).$$

In either case, fixing n at a constant and h_n either at a constant or a recursively estimated value implements an exponentially windowed version of the kernel estimator. (Similarly, one can phrase this in terms of θ as was done with the mean; see (Wegman and Marchette, 2003). These can in turn be used to estimate the "normal" activity of various measurements on the network, and provide a mechanism for detecting changes from normal, which in turn may indicate attacks. More information on these issues can be found in (Wegman and Marchette, 2003).

Similar approaches can be implemented for other density estimation techniques. In particular, the adaptive mixtures approach of (Pribe, 1994) has a simple recursive formulation that can be adapted to streaming data.

There are several applications of density estimation to intrusion detection that one might consider. It is obvious that unusually large downloads (or uploads) may be suspicious in some environments. While it is not clear that density estimation is needed for this application, there might be some value in detecting changes in upload/download behavior. This can be detected through the tracking of the number of bytes transferred per session.

Perhaps a more compelling application is the detection of trojan programs. A trojan is a program that appears to be a legitimate program (such as a telnet server) but acts maliciously, for example to allow access to the computer by unauthorized users. Obviously the detection of trojans is an important aspect of computer security.

Most applications (web, email, ftp, et cetera) have assigned ports on which they operate. Other applications may choose to use fixed ports, or may choose any available port. Detecting new activity on a given port is a simple way to detect a trojan program. More sophisticated trojans will replace a legitimate application, such as a web server. It is thus desirable to determine when a legitimate application is acting in a manner that is unusual.

Consider Fig. 5.2. We have collected data for two applications (web and secure shell) over a period of 1 hour, and estimated the densities of the packet length and inter arrival times. As can be seen, the two applications have very different patterns for these two measures. This is because they have different purposes: secure shell is a terminal service which essentially sends a packet for every character typed (there is also a data transfer mode to secure shell, but this mode was not present in these data); web has a data transfer component with a terminal-like user interaction.

By monitoring these and other parameters, it is possible to distinguish between many of the common applications. This can then be used to detect when an application is acting in an unusual manner, such as when a web server is being used to provide telnet services. See (Early and Brodley, 2003) for a more extensive discussion of this.

Note that web traffic has two main peaks at either end of the extremes in packet size. These are the requests, which are typically small, and the responses, which are pages or images and are broken up into the largest packets possible. The mass between the peaks mostly represent the last packets of transfers which are not a multiple of the maximum packet size, and small transfers that fit within a single packet.

The inter packet arrival times for secure shell also have two peaks. The short times correspond to responses (such as the response to a directory list command) and to characters typed quickly. The later bump probably corresponds to the pauses between commands, as the user processes the response. These arrival times are very heavy tailed because of the nature of secure shell. Sessions can be left open indefinitely, and if no activity occurs for a sufficiently long time, “keep alive” packets are sent to ensure that the session is still valid.

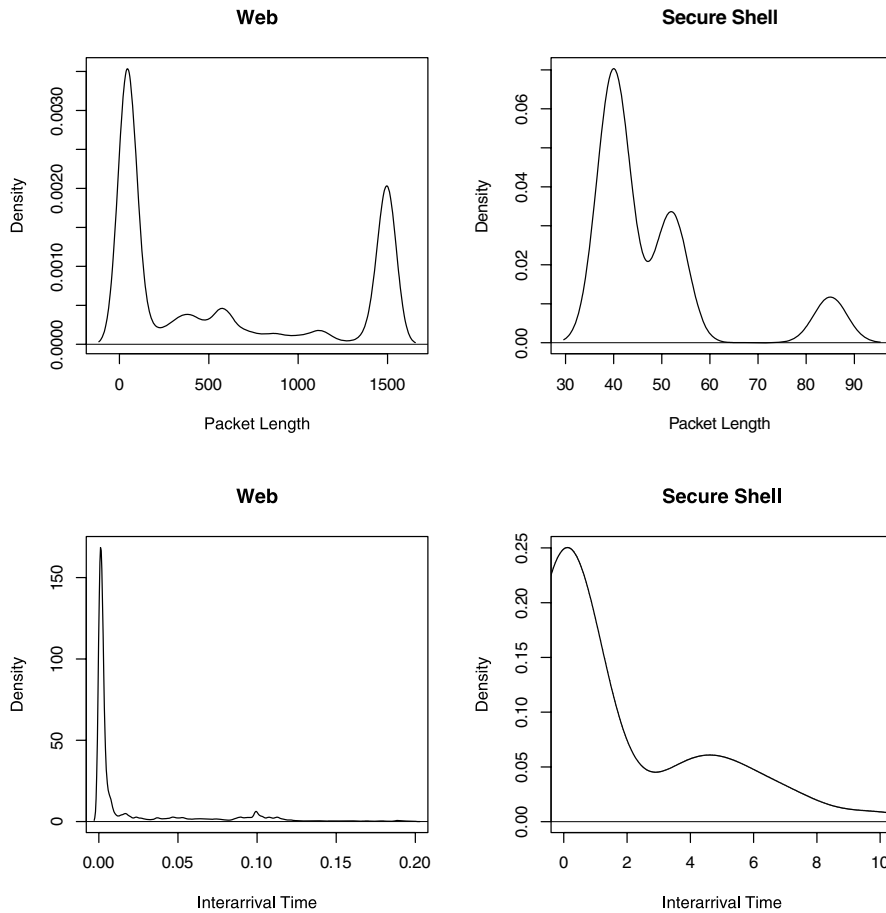


Figure 5.2. Packet length in bytes (*top*) and packet inter arrival times in seconds (*bottom*) for web (*left*) and secure shell (*right*) sessions. Kernel estimators were used to estimate the densities. The inter arrival times were truncated to show the bulk of the data

In (Early and Brodley, 2003) it is shown, in fact, that differences in the counts for the TCP flags can be used to differentiate applications. These, combined with mean inter packet arrival times and packet lengths (all computed on a window of n packets for various values of n), do a very creditable job of distinguishing applications. This is clearly an area in which recursive methods like those mentioned above would be of value. It also is reasonable to hypothesize that estimating densities, rather than only computing the mean, would improve the performance.

By detecting changes in the densities of applications it may be possible to detect when they have been compromised (or replaced) by a trojan program. It may also be possible to detect programs that are not performing as advertised (web servers acting like telnet servers, for example).

Visualization

Visualization of complex data is important but difficult. This is especially true of streaming data. While many complex techniques for visualization have been developed, simple scatter plots can be used effectively, and should not be shunned.

Figure 5.3 shows a scatter plot of source port against time for an 8 hour period of time. These are all the SYN packets coming in to a class B network (an address space of 65,536 possible IP addresses). This graphic, while simple, provides quite a few interesting insights.

Note that there are a number of curves in the plot. These are a result of the fact that each time a client initiates a session with a server, it chooses a new source port, and this corresponds to the previous source port used by the client incremented by one. Contiguous curves correspond to connections by a single source IP. Vertical gaps in the curves indicate that the IP visited other servers between visits to the network. It is also easy to see the start of the work day in this plot, indicated by the heavy over plotting on the right hand side.

The source ports range from 1024 to 65,536. Different applications and operating systems select ports from different ranges, so one can learn quite a bit from investigating plots like this.

The plot of Fig. 5.3 is static. Figure 5.4 is meant to illustrate a dynamic plot. This is analogous to the waterfall plots used in signal processing. It displays a snapshot in time that is continuously updated. As new observations are obtained they are plotted on the right, with the rest of the data shifting left, dropping the left most column. Plots like this are required for streaming data.

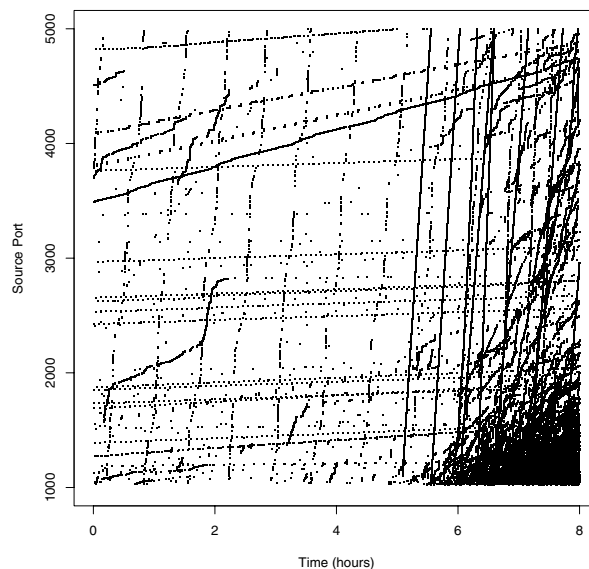


Figure 5.3. Source port versus time for all the incoming SYN packets for an 8 hour period

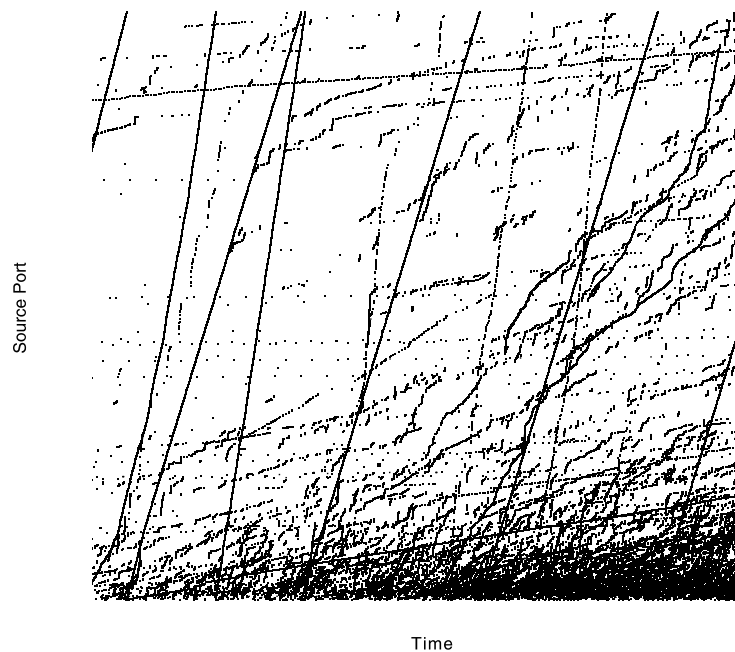


Figure 5.4. Source port versus time for a short time period, the last two hours from Fig. 5.3. As time progresses, the plot shifts from right to left, dropping the left most column and adding a new column on the right

Simple plots can also be used to investigate various types of attacks. In Fig. 5.5 is plotted spoofed IP address against time for a denial of service attack against a single server. Each point corresponds to a single unsolicited SYN/ACK packet received at the sensor from a single source. This plot provides evidence that there were actually two distinct attacks against this server. The left side of the plot shows a distinctive stripped pattern, indicating that the spoofed IP addresses have been selected in a systematic manner. On the right, the pattern appears to be gone, and we observe what looks like a random pattern, giving evidence that the spoofed addresses are selected at random (a common practice for distributed denial of service tools). Between about 0.03 and 0.06 there is evidence of overlap of the attacks, indicating that this server was under attack from at least two distinct programs simultaneously.

Another use of scatter plots for analysis of network data is depicted in Fig. 5.6. These data were collected on completed sessions. The number of packets is plotted against the number of bytes. Clearly there should be a (linear) relationship between these. The interesting observation is that there are several linear relationships. This is similar to the observations made about Fig. 5.2, in which it was noted that different applications use different packet lengths.

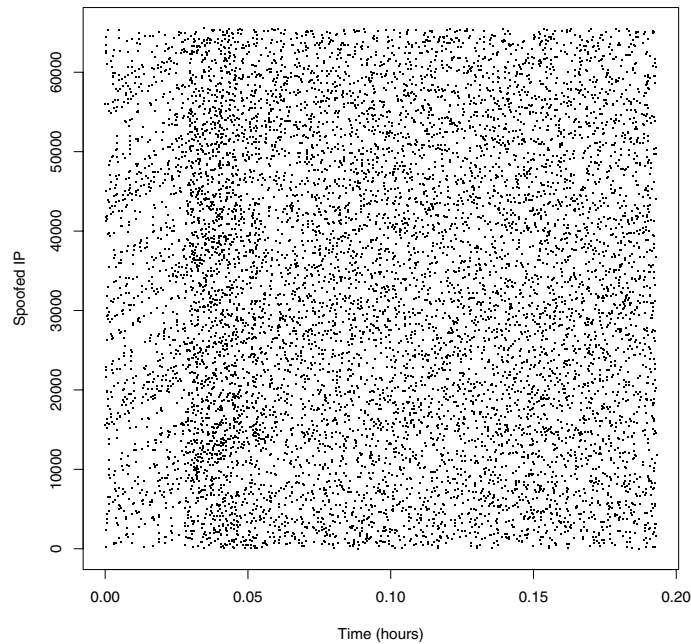


Figure 5.5. Plot of spoofed IP address against time for backscatter packets from a denial of service attack against a single server. The IP addresses have been converted to 16-bit numbers, since in this case they correspond to the final two octets of the IP address

Figure 5.7 shows the number of bytes transferred within a session plotted against the start time of the session. There is a lot of horizontal banding in this plot, corresponding mostly to email traffic. It is unknown whether the distinctive repetitive patterns are a result of spam (many email messages all the same size) or whether there are other explanations for this. Since these data are constructed from packet headers only, we do not have access to the payload and cannot check this hypothesis for these data. Figure 5.8 shows a zoom of the data. The band just below 400 bytes correspond to telnet sessions. These are most likely failed login attempts. This is the kind of thing that one would like to detect. The ability to drill down the plots, zooming and selecting observations to examine the original data, is critical to intrusion detection.

High dimensional visualization techniques are clearly needed. Parallel coordinates is one solution to this. In Fig. 5.9 we see session statistics for four different applications plotted using parallel coordinates.

One problem with plots like this is that of over plotting. Wegman solves this via the use of color saturation (see (Wegman and Dorfman, 2001, Wilhelm et al., 1999)). Without techniques such as this it is extremely difficult to display large amounts of data. Figure 5.9 illustrates this problem in two ways. First, consider the secure shell data in the upper left corner. It would be reasonable to conclude from this plot that secure shell sessions are of short duration, as compared with other

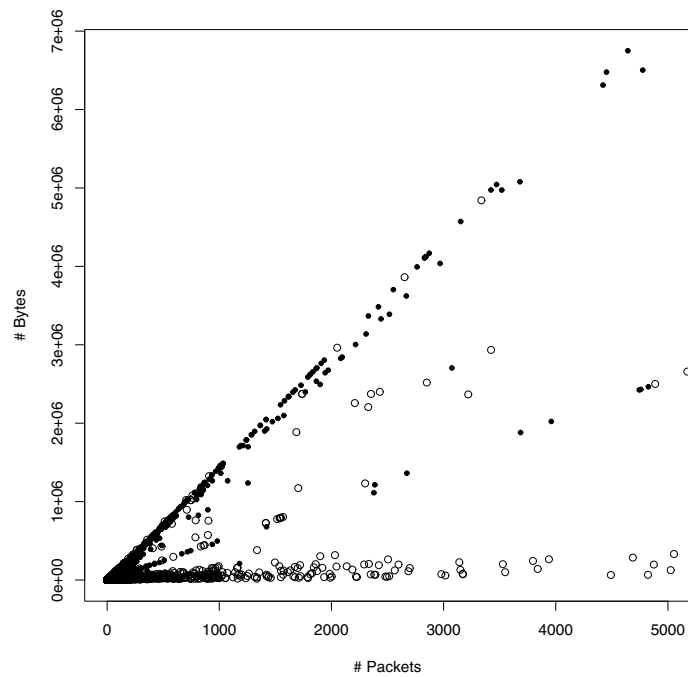


Figure 5.6. Number of bytes transferred within a completed session plotted against the number of packets within the session. *Solid dots* correspond to email sessions, *circles* correspond to all other applications

sessions. This is an artifact of the data. For these data there are only 10 secure shell sessions, and they all happen to be of short duration. Thus, we really need to look at a lot of data to see the true distribution for this applications. Next, look at the email plot in the upper right. Most of the plot is black, showing extensive over plotting. Beyond the observation that these email sessions have heavy tails in the size and duration of the sessions, little can be gleaned from this plot.

A further point should be made about the web sessions. Some of the sessions which are relatively small in terms of number of packets and bytes transferred have relatively long durations. This is a result of the fact that often web sessions will not be closed off at the end of a transfer. They are only closed when the browser goes to another web server, or a time-out occurs. This is an interesting fact about the web application which is easy to see in these plots.

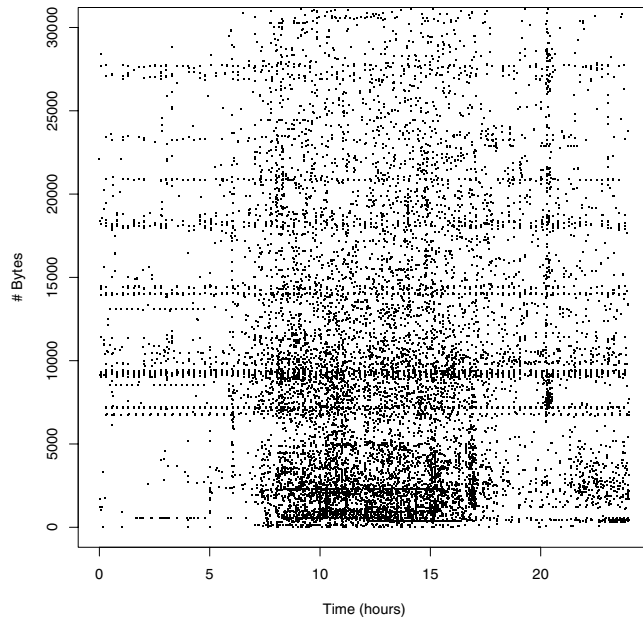


Figure 5.7. Number of bytes transferred for each session plotted against the starting time of the session for a single day

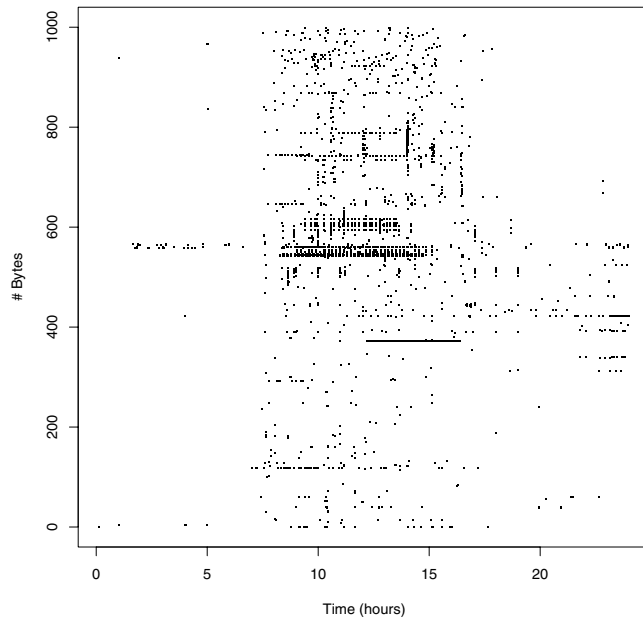


Figure 5.8. The portion of the sessions in Fig. 5.7 which were less than 1000 bytes

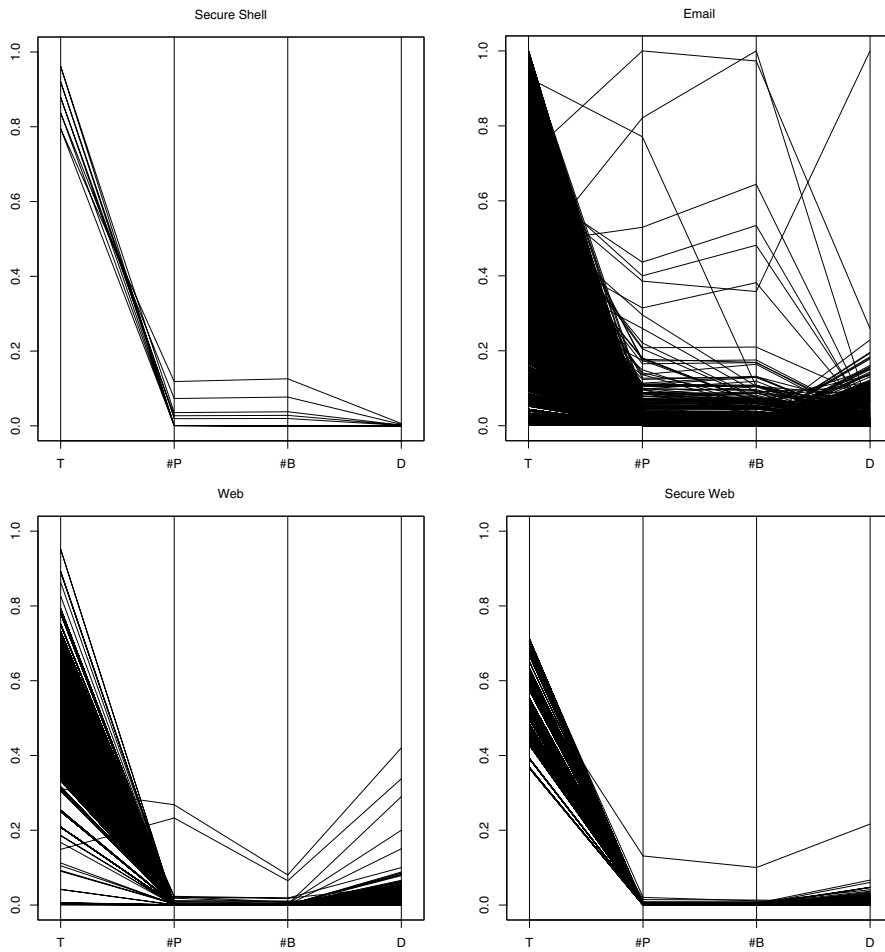


Figure 5.9. Parallel coordinates plots of session statistics for four different applications. From left to right, top to bottom they are: secure shell, email, web and secure web. The coordinates are the time of the initiating SYN packet, the total number of packets, the total number of bytes sent and the duration of the session. The axes are all scaled the same among the plots

5.6

Profiling and Anomaly Detection

We will now briefly consider host based intrusion detection. While the data considered is not network data, the statistical techniques used are applicable to network problems, as will be discussed.

One of the important problems of computer security is user authentication. This is usually performed by requiring the user to type a password at the initial login. Once a user is logged in, there are generally no checks to ensure that the person using the terminal is still the authorized person. User profiling seeks to

address this by extracting “person specific” information as the user interacts with the computer. By comparing the user’s activity with a profile of the user, it is hoped that masqueraders can be detected and locked out before they are able to do any damage.

We will discuss the usual host-based user profiling problem first, and then discuss a network based profiling application that has a similar flavor. The mathematics and statistics used for the two problems are very similar, only the data are different.

Several attempts have been made on this problem. Early work focused on utilizing keystroke timings. It was hoped that people had characteristic patterns of typing that could be discovered through measurement of the time between keystrokes for words or phrases. See for example (Bleha et al., 1990, Obaidat and Sadoun, 1997, Lin, 1997, Robinson et al., 1998).

This type of approach has been applied at the network level to crack passwords. (Song et al., 2001) describes using simple statistical techniques applied to packet arrival timings to determine the length of passwords in secure shell, and even to allow for the cracking of passwords. Secure shell is an application that allows remote login via an encrypted pathway. It sends a packet for each character typed, to minimize the delay for the user. Thus, by timing the packets, one can get an idea of what key combinations are being sent (it takes longer to type two characters with the same finger than it does if the characters are typed by fingers on different hands, for example). By utilizing statistics such as these, the authors were able to show that they could dramatically reduce the search space needed to crack the passwords.

Other work focuses on tracking user commands. The idea is that the command streams that users type (ignoring the arguments to the commands) could be used to authenticate the user in much the same way that keystroke timings could. A good discussion of this for statisticians can be found in (Schonlau et al., 2001). See also (Maxion, 2003, Maxion and Townsend, 2002) for some critiques of this work and extensions. The former paper considers arguments to the commands as well.

For Microsoft Windows operating systems, user command sequences are generally not applicable. Instead, window titles may be used. These correspond roughly to the same information that is contained in the Unix command lines. They typically contain the application name and the arguments to the applications such as the file open, the email subject, the web page visited, et cetera.

To illustrate this, we consider a set of data taken from six users on seven Windows NT machines over a period of several months. All window titles generated from the login to the logout were retained for each user/host pair (only one of the users was observed on a second host). Each time a window became active it was recorded. These data are a subset of a larger set. More information on these data, with some analysis of the data and performance of various classifiers can be found in (DeVault et al., 2003).

Table 5.1 shows some statistics on these data. Three sessions are shown for each user/host pair. The length of the login session (in seconds), the name of the first

Table 5.1. Session statistics for three login sessions for each user/host pair

User	Session	Login Length	1st App	Last App	#Apps	#Wins	#Titles
user1-host19	3	30,794	msoffice	msoffice	6	13	134
user1-host19	5	28,788	msoffice	msoffice	8	15	194
user1-host19	6	19,902	msoffice	msoffice	10	25	267
user1-host5	1	3472.47	explorer	explorer	3	6	34
user1-host5	2	142.98	explorer	explorer	2	3	6
user1-host5	40	21,912.79	explorer	explorer	7	25	187
user19-host10	5	31,432.5	msoffice	msoffice	7	8	133
user19-host10	6	16,886.3	msoffice	msoffice	6	7	75
user19-host10	11	2615.55	msoffice	acrord32	6	8	45
user25-host4	2	28,362.82	explorer	explorer	4	19	382
user25-host4	3	45,578.82	explorer	explorer	5	16	316
user25-host4	12	6788.44	explorer	explorer	4	11	102
user4-host17	10	19,445.96	wscript	explorer	8	21	452
user4-host17	30	6310.72	explorer	explorer	3	5	60
user4-host17	44	17,326.21	explorer	winword	8	10	138
user7-host20	10	23,163.6	outlook	outlook	5	7	51
user7-host20	11	44,004.11	wscript	mapisp32	5	5	72
user7-host20	12	33,125.27	wscript	outlook	5	7	166
user8-host6	1	31,395.08	wscript	explorer	7	14	116
user8-host6	4	1207.84	outlook	explorer	4	4	14
user8-host6	21	134.01	cmd	explorer	3	4	13

and last applications used within the session, and the number of distinct applications, windows and window titles are shown. The task is to extract statistics from a completed login session that allow one to determine whether the user was the authorized user indicated by the userid. This is an easier problem than masquerader detection, in which one tries to detect the masquerader (or authenticate the user) as the session progresses, and it is not assumed that the entire session corresponds to a single user (or masquerader).

The table indicates that there is some variability among the sessions of individual users, and this is born out by further analysis. Table 5.2 shows the most common window titles. The number of times the title occurs in the data set, the number of login sessions in which the title occurs, and the title itself are shown. Some words in the titles have been obfuscated by replacement with numbers in double brackets, to protect the privacy of the users. All common application and operating system words were left alone. The obfuscation is consistent across all sessions: there is a bijection between numbers and words that holds throughout the data.

Figure 5.10 shows part of a single login session. The rows and columns correspond to the list of words (as they appear in the titles) and a dot is placed where the word appears in both the row and column. The blocks of diagonal lines are characteristic of a single window in session. The “plus” in the lower left corner

Table 5.2. Window title usage

#	#Sessions	Window Title
7002	425	Inbox - Microsoft Outlook
2525	411	Program Manager
2188	215	Microsoft Word
792	126	Netscape
704	156	Print
672	213	Microsoft Outlook
639	156	<< 12761 >> << 9227 >>
592	170	<< 16193 >> - Message (<< 16184 >> << 5748 >>)
555	174	<< 6893 >> << 13916 >>
414	297	Microsoft(<< 3142 >>) Outlook(<< 3142 >>) << 7469 >>
413	36	<< 13683 >> << 3653 >> - Microsoft Internet Explorer
403	33	<< 13683 >> << 10676 >> - Microsoft Internet Explorer
402	309	- Microsoft Outlook
401	61	Microsoft PowerPoint
198	84	http:// << 1718 >>.<< 7267 >>.<< 4601 >> / << 16345 >>

shows a case of the user switching windows, then switching back. This type of behavior is seen throughout the data.

Many features were extracted from the data, and several feature selection and dimensionality reduction techniques were tried. The results for these approaches were not impressive. See (DeVault et al., 2003) for more discussion.

The classifiers that worked best with these data were simple intersection classifiers. For each session, the total set of window titles used (without regard to order) was collected. Then to classify a new session, the intersection of its title set with those from user sessions was computed, and the user with the largest intersection was deemed to be the user of the session. Various variations on this theme were tried, all of which performed in the mid to high 90 percent range for correct classification.

Much more needs to be done to produce a usable system. Most importantly, the approach must move from the session level to within-session calculations. Further, it is not important to classify the user as one of a list of users, but to simply state whether the user's activity matches that of the userid. It may be straight forward to modify the intersection classifier (for example, set a threshold and if the intersection is below the threshold, raise an alarm) but it is not clear how well this will work.

We can state a few generalities about user profiling systems. Users are quite variable, and such systems tend to have an unacceptably high false alarm rate. Keystroke timings tend to be much more useful when used with a password or pass phrase than in free typing. No single technique exists which can be used reliably to authenticate users as they work.

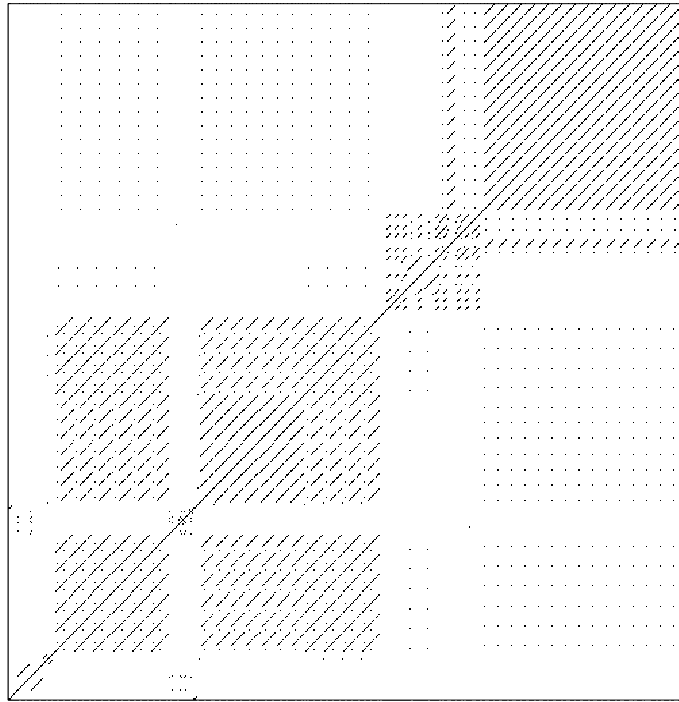


Figure 5.10. First 500 words from a single session. The rows and columns correspond to words in the order in which they appear (with duplicates). A dot is plotted in (i, j) if the same word is in row i and column j

The intersection classifier leads to interesting statistics. We can construct graphs using these intersections, each node of the graph corresponding to a session, with an edge between two nodes if their sets intersect nontrivially (or have an intersection of size at least T).

In another context (profiling the web server usage of users) (Marchette, 2003) discusses various analyses that can be done on these graphs. This uses network data, extracting the source and destination IP addresses from the sessions. In these data there is a one-to-one correspondence between source IP address and user, since all the machines considered were single user machines.

In this case the nodes correspond to users and the sets consist of the web servers visited by the user within a period of a week. A random graph model, first described in (Karonski et al., 1999) is used as the null hypothesis corresponding to random selection of servers. The model assumes a set \mathcal{S} of servers from which the users draw. To define the set of servers for a given user, each server is drawn with probability p . Thus, given the observations of the sets S_i drawn by the users, we must estimate the two parameters of the model: $m = |\mathcal{S}|$ and p . These can be estimated using maximum likelihood (see also (Marchette, 2004) for discussion of this and other types of intersection graphs). With the

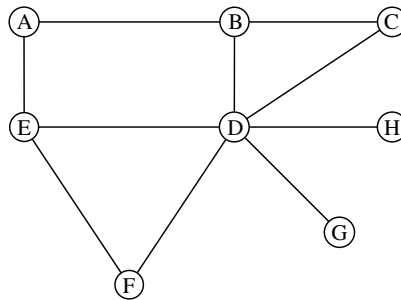


Figure 5.11. A graph of the users with significantly large intersections. The edges for which the intersection size was statistically significant for 95% of the weeks are shown

notation

$$k_i = |S_i|$$

$$M_i = \left| \bigcup_{j=1}^i S_j \right|$$

$$u_i = M_i - M_{i-1} ,$$

the likelihood is easily shown to be

$$L = \prod_{j=1}^n \binom{M_{j-1}}{k_j - u_j} \binom{m - M_{j-1}}{u_j} p^{k_j} (1 - p)^{m - k_j} .$$

Using data collected for several months, (Marchette, 2003) computed the probability of any given edge, under the null hypothesis, and retained those that had a significantly large intersection (after correcting for the multiple hypotheses tested). The most common of these were retained, and the resulting graph is shown in Fig. 5.11.

There are two triangles in Fig. 5.11, and it turns out that the users in these correspond to physicists working on fluid dynamics problems. Users A, D and E are system administrators. Thus, there is some reason to believe that the relationships we have discovered are interesting.

The model is simplistic, perhaps overly so. It is reasonable to assume that users have different values of p , and some preliminary investigation (described in (Marchette, 2003)) bears this out. This is an easy modification to make. Further, intuition tells us that perhaps all web servers should not have the same probabilities either. This is more problematic, since we cannot have a separate probability for each server and hope to be able to estimate them all. A reasonable compromise might be to group servers into common/rare groups or something similar.

The above discussion illustrates one of the methodologies used for anomaly detection. For determining when a service, server, or user is acting in an unusual

manner, one first groups the entities using some model, then raises an alert when an entity appears to leave the group. Alternatively, one can have a single entity, for example “the network” or a given server, and build a model of the behavior of that entity under normal conditions. When the behavior deviates from these conditions by a significant amount, an alert is raised.

Other researchers have investigated the profiling of program execution, for the purpose of detecting attacks such as buffer overflows which can cause the program to act in an unusual way. See for example (Forrest et al., 1994, Forrest et al., 1997, Forrest and Hofmeyr, 1998, Tan and Maxion, 2002). Programs execute sequences of system calls, and the patterns of system calls that occur under normal conditions are used to detect abnormal execution.

5.7 Discussion

There are many areas in which computational statistics can play a part in network intrusion detection and other security arenas. We have seen a few in this chapter, including modeling denial of service attacks, visualization, the analysis of streaming data applied to network data and profiling and anomaly detection.

The biggest problems for intrusion detection systems are the false alarm rates and the detection of novel attacks. The enormous amount of data that must be processed requires that false alarm rates must be extremely low. Typical network data consists of millions of packets an hour, and system administrators generally do not have time to track down more than a few false alarms a day. Signature based systems have the advantage that they rarely false alarm (assuming the signature is properly defined), but they tend to have poor performance on novel attacks. Thus it is essential that techniques be found that detect novelty that is “bad” without alarming on novelty that is benign.

One area we have not discussed is modeling attack propagation. Early work on this can be found in (Kephart and White, 1991, Kephart and White, 1993). See also (Wierman and Marchette, 2004) for a related model. For a discussion of the slammer worm, see <http://www.cs.berkeley.edu/~nweaver/sapphire/>. The slammer worm was interesting because the spread was self-limiting: the worm spread so fast that the available bandwidth was reduced to the point that the worm was unable to continue to spread at its initial rate. Models for these types of worms is an interesting area of study.

References

- Amoroso, E. (1999). *Intrusion Detection: An Introduction to Internet Surveillance, Correlation, Trace Back, Traps, and Response*. Intrusion.net Books, Sparta, New Jersey.
- Anderson, D., Lunt, T. F., Javitz, H., Tamaru, A., and Valdes, A. (1995). Detecting unusual program behavior using the statistical component of the next-generation

- intrusion detection expert system (nides). Technical Report SRI-CSL-95-06, SRI International.
- anonymous (1997). *Maximum Security*. Sams.net Publishing, Indianapolis, IN.
- Bace, R. G. (2000). *Intrusion Detection*. MacMillan Technical Publishing, Indianapolis.
- Bleha, S., Slivinsky, C., and Hussien, B. (1990). Computer-access security systems using keystroke dynamics. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(12):1217–1222.
- DeVault, K., Tucey, N., and Marchette, D. (2003). Analyzing process table and window title data for user identification in a windows environment. Technical Report NSWCDD/TR-03/122, Naval Surface Warfare Center.
- Early, J. P. and Brodley, C. E. (2003). Behavioral authentication of server flows. In *The 19th Annual Computer Security Applications Conference*. to appear.
- Escamilla, T. (1998). *Intrusion Detection: Network Security Beyond the Firewall*. John Wiley & Sons, Inc., New York.
- Forrest, S. and Hofmeyr, S. A. (In press). Immunology as information processing. In Segel, L. A. and Cohen, I., editors, *Design Principles for the Immune System and Other Distributed Autonomous Systems*, Santa Fe Institute Studies in the Sciences of Complexity. Oxford University Press, Oxford, UK. Also available at www.cs.unm.edu/~forrest/ism_papers.htm.
- Forrest, S., Hofmeyr, S. A., and Somayaji, A. (1997). Computer immunology. *Communications of the ACM*, 40:88–96.
- Forrest, S., Perelson, A. S., Allen, L., and Cherukuri, R. (1994). Self-nonsel self discrimination in a computer. In *1994 IEEE Symposium on Research in Security and Privacy*. Also available at www.cs.unm.edu/~forrest/isa_papers.htm.
- Giles, K., Marchette, D. J., and Priebe, C. E. (2003). A backscatter characterization of denial-of-service attacks. In *Proceedings of the Joint Statistical Meetings*. to appear.
- Karonski, M., Singer, K., and Scheinerman, E. (1999). Random intersection graphs: the subgraph problem. *Combinatorics, Probability and Computing*, 8:131–159.
- Kephart, J. O. and White, S. R. (1991). Directed-graph epidemiological models of computer viruses. In *Proceedings of the IEEE Computer Society Symposium on Research in Security and Privacy*, pages 343–359.
- Kephart, J. O. and White, S. R. (1993). Measuring and modeling computer virus prevalence. In *Proceedings of the IEEE Computer Society Symposium on Research in Security and Privacy*, pages 2–15.
- Lin, D.-T. (1997). Computer-access authentication with neural network based keystroke identity verification. In *International Conference on Neural Networks*, pages 174–178.
- Marchette, D. J. Passive detection of denial of service attacks on the internet. In Chen, W., editor, *Statistical Methods in Computer Security*. Marcel Dekker. to appear.
- Marchette, D. J. (2001). *Computer Intrusion Detection and Network Monitoring: A Statistical Viewpoint*. Springer, New York.

- Marchette, D. J. (2002). A study of denial of service attacks on the internet. In *Proceedings of the Army Conference on Applied Statistics*, pages 41–60.
- Marchette, D. J. (2003). Profiling users by their network activity. In *Proceedings of the Joint Statistical Meetings*. to appear.
- Marchette, D. J. (2004). *Random Graphs for Statistical Pattern Recognition*. John Wiley & Sons, New York.
- Maxion, R. A. (2003). Masquerade detection using enriched command lines. In *International conference on dependable systems and networks(DNS-03)*. IEEE Computer Society Press.
- Maxion, R. A. and Townsend, T. N. (2002). Masquerade detection using truncated command lines. In *International conference on dependable systems and networks(DNS-02)*. IEEE Computer Society Press.
- Moore, D., Voelker, G. M., and Savage, S. (2001). Inferring Internet denial-of-service activity. Available on the web at www.usenix.org/publications/library/proceedings/sec01/moore.html. USENIX Security '01.
- Northcutt, S., Novak, J., and McLaclan, D. (2001). *Network Intrusion Detection. An Analyst's Handbook*. New Riders, Indianapolis.
- Obaidat, M. S. and Sadoun, B. (1997). Verification of computer users using keystroke dynamics. *IEEE Transactions on Systems, Man, and Cybernetics*, 27(2):261–269.
- Priebe, C. E. (1994). Adaptive mixture density estimation. *Journal of the American Statistical Association*, 89:796–806.
- Proctor, P. E. (2001). *The Practical Intrusion Detection Handbook*. Prentice-Hall, Englewood Cliffs, NJ.
- Robinson, J. A., Liang, V. M., Chambers, J. A. M., and MacKenzie, C. L. (1998). Computer user verification using login string keystroke dynamics. *IEEE Transactions on Systems, Man, and Cybernetics*, 28(2):236–241.
- Schonlau, M., DuMouchel, W., Ju, W.-H., Karr, A. F., Theus, M., and Vardi, Y. (2001). Computer intrusion: Detecting masquerades. *Statistical Science*, 16:58–74.
- Song, D. X., Wagner, D., and Tian, X. (2001). Timing analysis of keystrokes and timing attacks on SSH. In *Proceedings of the 10th USENIX Security Symposium*. <http://www.usenix.org/publications/library/proceedings/sec01/song.html>.
- Stevens, W. R. (1994). *TCP/IP Illustrated, Volume 1: The Protocols*. Addison-Wesley, Reading.
- Tan, K. M. C. and Maxion, R. A. (2002). “why 6?” defining the operational limits of stide, an anomaly-based intrusion detector. In *IEEE Symposium on Security and Privacy*. IEEE Computer Society Press.
- Wegman, E. J. and Davies, H. I. (1979). Remarks on some recursive estimators of a probability density. *Annals of Statistics*, 7:316–327.
- Wegman, E. J. and Dorfman, A. (2001). Visualizing cereal world. Technical Report TR 178, George Mason University, Center for Computational Statistics.
- Wegman, E. J. and Marchette, D. J. (2003). On some techniques for streaming data: a case study of Internet packet headers. *JCGS*, 12(4):893–914.

- Wierman, J. C. and Marchette, D. J. (2004). Modeling computer virus prevalence with a susceptible-infected-susceptible model with reintroduction. *Computational Statistics and Data Analysis*, 45(1):3–23.
- Wilhelm, A. F. X., Wegman, E. J., and Symanzik, J. (1999). Visual clustering and classification: The oronsay particle size data set revisited. *Computational Statistics*, pages 109–146.
- Yamato, H. (1971). Sequential estimation of a continuous probability density function and the mode. *Bulletin of Mathematical Statistics*, 14:1–12.

Index

- (t, m, s) -net, 51
- 2×2 table, 779
- L_1 -fit, 689
- L_1 -minimization, 688
- L_1 -regression, 684
- Q-function, 142, 143, 148
 - complicated E-step, 154
 - generalized EM (GEM) algorithm, 142
 - MC approximation, 155
- α -channel, 307
- α -level contour, 307
- α -outlier, 672
- α -outlier region, 672, 686
- α -quantile, 669
- π^* -irreducible, 76
- 3D Visual Data Mining (3DVDM) System, 317

- abbreviation method, 393
- acceptance rate, 80
 - empirical, 742
- acceptance-complement method, 62
- access specifier, 411
- accumulated proportion, 568
- adaptive mixtures, 1034
- adaptive SPSA, 186
- adaptivity, 754, 757
 - invalid, 754
- add-with-carry, 49
- additive models, 557, 704
- address spoofing, 1032
- adjusted dependent variable, 600
- aesthetics, 363

- affine equivariance, 673, 680
- affine transformation, 662
- AICc, 444, 452
- aircraft design, 184
- Akaike's information criterion (AIC), 438, 444, 449, 450, 452, 453, 456, 602, 628, 728, 846
- algebra, 343
- algebraic curve fitting, 575
- algebraic surface fitting, 575
- alias method, 61
- allowable splits, 822
- alternating expectation-conditional maximization (AECM) algorithm, 160, 164
- Amdahl's law, 242
- anaglyphs, 313, 314
- Analysis of Functional Neuroimages (AFNI), 1021
- analysis of variance, 686
- Andrews plot, 306
- annealed entropy, 847
- anomaly detection, 1030
- antithetic variables, 965
- antithetic variates, 500
- aperiodic, 76
- applications, 870
- approximate distance, 576
- approximations, 779
- AR model, 723, 728, 751
 - order, 723
- ArcView/XGobi, 316
- ArcView/XGobi/XploRe, 318

- arithmetic mean, 661
- artificial intelligence, 187
- asset returns, 952, 961
- association rules, 795
- asymptotic α -confidence interval, 669
- asymptotic bias, 548
- asymptotic distribution, 184
- asymptotic normality, 675
- asymptotic relative efficiency, 658, 771
- asymptotic variance, 549, 660
- asymptotically random, 52
- attack propagation, 1048
- autocorrelation plots, 84
- autocorrelation time, 78
- automatic methods, 60, 64
- auxiliary variables, 92
- averaged shifted histogram (ASH), 307, 312, 528

- backscatter, 1033, 1039
- bandwidth, 524, 541, 543, 547
- bar chart, 307
- base class, 413
- base-line intensity, 779
- batch means, 78
- Bayes
 - optimal, 844
 - theorem, 728
- Bayes factor, 438, 444, 450, 457–459, 725, 972
 - approximation, 737, 739
 - computation, 729
- Bayesian
 - hierarchical structures, 743
 - software, 763
- Bayesian classifiers, 794
- Bayesian framework, 138
 - Gibbs sampler, 162–165
 - MAP estimate, 162, 163
 - MCMC, 162, 163
- Bayesian inference, 955, 958–960, 962, 965, 969, 972, 975
- Bayesian information criterion (BIC), 438, 444, 449, 450, 452, 456, 459, 603
- Bayesian statistics, 72, 720
- Beowulf class cluster, 241

- Bernoulli data, 592
- Bertillon, Alphonse, 743
- BFGS algorithm, 967
- bias, 547, 659, 666, 668, 671, 676, 679
 - function, 679
 - functional, 676
- bias estimation, 553
- bias-variance tradeoff, 444, 446, 447, 547
- binary data, 592
- binary representation, 190
- binary response data, 83
- Binary Response Model, 713
- binary search, 60
- binary tree, 815
- binomial distribution, 594, 726
- bioinformatics, 164
- bioprocess control, 184
- birth-and-death process, 749
- birthday spacings test, 57
- bisquare function, 541
- bivariate histogram, 521
- blind random search, 177
- block bootstrap, 484, 486, 488, 491
- block design, 1009
- blocks, 73
- blood oxygen level dependent (BOLD) contrast, 1007
- Bonferonni correction, 1020
- boosting, 853
- bootstrap, 152, 468–492, 501, 556, 835
 - nonparametric, 153
 - smooth, 759
- bootstrap for dependent data, 470, 472, 483, 492
- Boston Housing data, 387, 393, 394
- boundary bias, 542
- bounded algebraic curve, 579
- boxplot, 657
- brain activation, 1008
- Brain Voyager, 1021
- breakdown, 658, 664, 676
- breakdown point, 660, 667, 668, 671, 676, 683, 684
- breakdown point of M-functional, 677
- Breslow–Peto method, 775, 776

- bridge estimator, 739
- bridge sampling, 738, 739
- brush-tour, 303
- brushing, 300, 388
- brushing with hue and saturation, 307
- burn-in, 79

- candidate generating density, 79
- canonical link, 595, 600
- cascade algorithm, 229
- Castillo–Hadi model, 772
- categorization, 607
- Cauchy distribution, 666
- CAVE Audio Visual Experience Automatic Virtual Environment (CAVE), 314
- censored data, 560, 829
- central limit theorem, 662, 730
- characteristic polynomial, 50, 53–55
- chi-squared-test, 602
- choice
 - brand, 952, 957
 - discrete, 952
 - probabilities, 954, 956, 960
 - transport mode, 952
- Cholesky decomposition, 105
- chromosome, 187, 188
- class, 407
- class diagram, 411
- class hierarchy, 413
- class interface, 411
- classification, 814, 843
- clustered data, 614
- clustering, 793, 973, 974
 - duration, 970
 - volatility, 961
- coefficient of determination, 628
- coefficient of variation, 770, 771
- collision test, 57
- color, 392
- color model, 306, 313
- combined generators, 47, 54
- command streams, 1043
- common cumulative mean function, 779
- common parameter, 783
- common random numbers, 500
- common trend parameter, 779
- complete-data
 - information matrix, 152
 - likelihood, 139, 140
 - log likelihood, 141, 142, 146–148, 150, 155, 158
 - problem, 139, 146, 148
 - sufficient statistics, 158
- completion, 744
- complexity, 847
- complexity parameter, 819
- composition method, 64
- COMPSTAT Conferences, 8
- computational effort, 730
- computational inference, 4, 12–14
- computational statistics, 5
- conditional likelihood, 777, 781, 783
- conditional tests, 478, 481–483
- Conditional VaR, 912
- conditioned choropleth maps (CCmaps), 321
- confidence, 801
- confidence interval, 468, 469, 472, 478–481, 554, 659, 669, 675
- confidence level, 726
- confidence regions, 725
- conjugate gradient method, 125
- consistency, 844–846
 - uniform convergence, 846
- consistent, 662
- constraints, 174, 192
- constructor, 409
- contingency table, 613
- continuum regression, 638
 - ridge regression, 639
- contour shell, 307
- contour surface, 307
- control variables, 965
- convergence, 141–143, 145, 156–158, 160
 - monotonicity property, 143, 154, 156, 160
 - speeding up, 159–162
- convergence assessment, 730
- convergence theory, 179, 182
- convolution method, 64
- coordinates, 362
- copula

- Archimedean, 943
- elliptical, 943
- estimation, 944
- simulation
 - – conditional distributions method, 944
- cost-complexity, 819
- count data, 98, 592
- count model, 975
- counting process, 779, 780
- covariance functional, 678
- covariate, 682, 774
- covering numbers, 847
- Cox model, 774, 776
- Cox's discrete model, 776
- critical sampling, 217
- cross, 344, 351, 368
- cross validation, 438, 444, 450, 453, 454, 456, 551, 630, 819
- crossover, 188, 189
- CrystalVision, 309–311, 317
- cumulative logit model, 613
- cumulative mean function, 780
- curse of dimension, 524, 557, 736, 741
- cyclic menu, 386

- data augmentation, 92, 955, 972, 975
- data mining (DM), 160, 295
- Data Viewer, 311, 312
- data visualization, 519
- DataDesk, 399
- dataflow, 339
- daughter node, 817
- DC shifts, 1011
- Decision theory, 721
- decision trees, 794, 796, 814
- decomposition algorithm, 232
- defensive sampling, 759
- degrees of freedom, 444, 453, 549
- denial of service attack, 1030, 1032, 1038, 1039
- density estimation, 560, 561
- dependent data, 470, 472, 483–485, 491, 492
- descriptive modeling, 793
- design matrix, 543
- design of experiments, 498
- destructor, 410

- deterministic annealing EM (DAEM) algorithm, 151
- deterministic simulation, 499, 511
- Deutscher Aktienindex (DAX), 929, 941
- deviance, 597
 - penalized, 728
- deviance information criterion (DIC), 728
- Diamond Fast, 309
- differentiable, 662, 669
- digamma function, 771
- dimension
 - high, 722, 754
 - matching, 748
 - unbounded, 724, 729
 - unknown, 748
- dimension reduction, 157, 159, 557
- Dimension reduction methods of explanatory variables, 584
- Dirichlet distribution, 975
- discrepancy, 40
- discrete logistic model, 774
- discrete optimization, 171, 186
- dispersion parameter, 601
- distributed memory, 239
- distribution
 - α -stable, 915, 916
 - – S parametrization, 917
 - – S^0 parametrization, 917
 - – characteristic function, 917
 - – density function, 918
 - – direct integration method, 919
 - – distribution function, 918
 - – Fama–Roll method, 922
 - – maximum likelihood estimation, 926
 - – method of moments, 923
 - – regression method, 924, 925
 - – simulation, 921
 - – STABLE program, 920, 927
 - binomial, 593, 726
 - Cauchy, 917
 - folded t , 735
 - Gaussian, 917
 - generalized hyperbolic
 - – density function, 935

- maximum likelihood estimation, 938, 939
- mean, 935
- simulation, 936
- variance, 935
- generalized inverse Gaussian (GIG), 933
 - simulation, 936, 937
- hyperbolic, 932
 - density function, 934
 - inverse, 941
- inverse Gaussian (IG)
 - simulation, 937
- Lévy, 917
- Lévy stable, 915
- mixture, 743, 749
- normal inverse Gaussian (NIG)
 - density function, 935
 - simulation, 937
 - tail estimates, 939
 - tails, 936
- predictive, 723
- proposal, 733
- stable Paretian, 915
- t , 732
- target, 740
- truncated stable (TLD), 930
 - characteristic function, 931
- dot plot, 519
- doubledecker plot, 810
- Dow Jones Industrial Average (DJIA), 913, 914, 928, 940
- downweighting outlying observations, 665
- dual lattice, 44, 52
- dynamic duration model/analysis, 952, 961, 969

- E-step (Expectation step), 139, 141, 583
 - exponential family, 141
 - factor analysis model, 158
 - failure-time data, 148
 - generalized linear mixed models (GLMM), 155
 - Monte Carlo, 154, 155
 - nonapplicability, 148
 - normal mixtures, 146
 - early binding, 420
 - effective number of parameters, 550
 - efficiency, 664, 670
 - efficiency of the sample mean, 771
 - eigenvalues, 126
 - inverse iterations, 129
 - Jacobi method, 127
 - LR method, 129
 - power method, 127
 - QR method, 128
 - eigenvectors, 126
 - electroencephalogram (EEG), 1006
 - elitism, 188, 189
 - EM algorithm, 745
 - extensions, 140, 160, 154
 - EM mapping, 142, 144
 - embarrassingly parallel, 244
 - empirical measure, 661
 - encapsulation, 406
 - encoding, 188
 - entropy, 57, 817
 - equidissection, 51
 - equidistribution, 51
 - equivariance, 662, 673
 - ergodic chain, 77
 - estimation vs. testing, 728
 - estimator
 - harmonic mean, 739
 - maximum a posteriori (MAP), 728
 - Euler's constant, 771
 - evolution strategies, 186
 - evolutionary computation, 186
 - exact distance, 576
 - excess kurtosis, 961, 968
 - expectation-conditional maximization (ECM)
 - algorithm, 156, 164
 - multicycle ECM, 156, 157
 - expectation-conditional maximization either (ECME) algorithm, 159, 160, 164
 - expected shortfall (ES), 912
 - expected tail loss (ETL), 912
 - EXPLOR4, 311
 - exploratory data analysis (EDA), 295, 788
 - exploratory spatial data analysis (ESDA), 317
 - ExplorN, 305, 309–311, 317

- exponential density function, 770
- exponential distribution, 672, 769, 770
- exponential family, 141, 147, 148, 592, 593, 596, 722
 - sufficient statistics, 141, 158
- Extensible Markup Language (XML), 340
- extreme value distribution, 959

- factor analysis model, 157
- failure-time data
 - censored, 140, 147
 - exponential distribution, 148
- false discovery rate (FDR), 1020
- fat-shattering dimension, 847
- fault detection, 185
- feedforward network, 800
- filter
 - high-pass, 223
 - quadrature mirror, 223
- final prediction error (FPE), 450
- finite mixture, 972
 - model, 952, 971, 972, 975
 - of Gaussian densities, 966
- finite-difference SA (FDSA), 181
- Fisher consistent, 660
- Fisher information, 781
 - generalized linear model (GLM), 602
- Fisher scoring algorithm, 599
- fitness function, 187, 189
- fitness proportionate selection, 189
- Fitts forecasting model, 390
- floating-point, 188
- focusing, 301
- font, 392
- fork-join, 241
- Fourier plot, 366
- Fourier space, 1010
- Fourier transform, 1010
- Fréchet differentiable, 669, 676, 679
- free-induction decay (FID) signal, 1004
- frequency domain bootstrap, 489–491
- frequency polygon, 528
- Friedman's index, 569
- full conditional distributions, 89
- full likelihood, 776

- full-screen view, 390
- Functional Image Analysis Software – Computational Olio (FIASCO), 1014, 1022
- functional model, 540
- functional neuroimaging, 1006

- gain sequences, 182
- gamma distribution, 594, 769, 970
- GARCH, 474–476, 478–481, 952, 961, 973, 974
- Gauss–Jordan elimination, 117
- Gauss–Newton method, 646
- Gauss–Seidel method, 122, 125
- Gaussian quadrature, 458
- Gaussian simulation smoother, 965, 966
- Gaussian/normal distribution, 955–963, 965–970, 973
 - Matrix, 956
 - truncated, 956
- gene expression data, 164
- generalized additive model, 616
- generalized cross validation, 438, 444, 453, 455, 456, 460, 552
- generalized degrees of freedom, 445, 450–452
- generalized EM (GEM) algorithm, 142, 156, 157
- generalized estimating equations (GEE), 615
- generalized feedback shift register (GFSR), 51, 54
- generalized linear mixed models (GLMM), 154
- generalized linear model (GLM), 592
- generalized maximum likelihood method, 453, 460, 775
- generalized method of moments, 954
- generalized partial linear model, 616
- generalized principal components, 571
- generalized principal components analysis (GPCA), 571
- genetic algorithm, 186, 191
- geographic brushing, 317
- Geometric distribution, 594
- geometrically ergodic, 77
- getter, 406
- GGobi, 309, 311–313, 399

- Gibbs sampling algorithm, 73
 Gibbs sampling/sampler, 162–165, 743, 747, 955, 957, 960, 966, 972, 973, 975
 – griddy-, 975
 – mixing of, 744
 Givens rotations, 110
 Glivenko–Cantelli theorem, 661
 global optimization, 180, 186
 global solutions, 175
 goodness of fit, 40, 444, 547, 555
 gradient approximation, 181, 183
 Gram–Schmidt orthogonalization, 112
 grand tour, 303, 305, 306
 graphics algebra, 343
 Green’s algorithm, 748
 Greenwood’s formula, 769
 gross error model, 676
 gross error neighbourhood, 667, 684
 Gumbel distribution, 958
 Gustafson’s law, 243
- Hall’s index, 570
 Halton sequences, 959
 Hampel identifier, 672
 hard thresholding, 443
 harmonic mean, 738
 hat function, 62
 hat matrix, 550
 Hawkes process, 971
 hazard function, 768, 779, 970
 hazard rate, 768
 head motion, 1015–1017
 Heisenberg’s uncertainty principle, 208
 Hessian (or Jacobian) matrix, 186
 Hessian (second derivative) matrix, 181
 heterogeneity, 955, 957, 958, 960, 970
 heterogeneous populations, 727, 744
 heteroscedasticity, 460
 hexagonal bins, 522
 hidden Markov model, 143, 164, 165
 hierarchical Bayes, 960
 hierarchical command sequence, 393
 high breakdown affine equivariant location and scale functionals, 677
 high breakdown regression functional, 685
 higher-order kernels, 524
 highest possible breakdown point, 668
 highest posterior region, 725
 Householder reflections, 109
 HPF (High Performance Fortran), 260
 Huber distribution, 659
 hue brushing, 307
 human-machine interaction, 185
 HyperVision, 310, 311
 hypotheses, 724
 hypothesis testing, 555
- i.i.d. resampling, 468, 473, 474, 485, 490
 identifiability, 723
 identification, 958, 972
 – problem, 972, 974
 – restrictions, 954, 972, 974
 Image Analysis, 164
 image grand tour (IGT), 305
 image registration, 1016
 immersive projection technology (IPT), 314, 316, 317
 importance function, 731, 963–965
 – choice of, 731
 – with finite variance, 731
 importance sampling, 81, 458, 500, 964, 965
 – and regular Monte Carlo, 734
 – degeneracy of, 737, 760
 – efficient (EIS), 962–965, 967–971
 – for model choice, 737
 incomplete-data
 – likelihood, 139, 141
 – missing data, 139, 140, 146, 148, 152, 155, 165
 – problem, 138, 139, 146, 163
 incremental EM (IEM) algorithm, 161
 independence M–H, 81
 independence of irrelevant alternatives (IIA), 958
 independent increments, 780
 indexed search, 61
 inefficiency factor, 78
 infinite collection of models, 729
 influence function, 659
 information criterion

- Akaike, 438, 444, 449, 450, 452, 453, 456, 602, 628, 728, 846
- Bayesian, 438, 444, 449, 450, 452, 456, 459, 603
- Schwarz, 628
- information matrix
 - complete-data, 152
 - expected, 151, 152
 - observed, 151, 152
- inheritance, 413, 418
- injected randomness, 174
- instance, 407
- integral, 952–954, 959, 960, 962, 963, 965
 - approximation, 729
 - high dimensional, 955
 - multiple, 962
 - ratio, 725
- integrated mean square error, 523
- intensity
 - function, 971
 - model, 969, 971
- intensity functions, 779
- inter arrival time, 1035, 1036
- interaction term, 609
- interface, 406, 419, 423
- interface for derivation, 411
- Interface Symposia, 8
- International Association for Statistical Computing (IASC), 8
- internet protocol, 1031
- intersection classifier, 1045, 1046
- intersection graph, 1046
- invariant, 74
- Inverse Gaussian distribution, 594
- inverse iterations, 129
- inverse moments, 185
- inversion method, 37, 59
- inverted gamma density/prior, 966, 969, 973
- inverted Wishart distribution, 956, 960
- iterative refinement, 119
- iterative simulation algorithms, 163
- iteratively reweighted least squares, 596, 598

- Jacobi method, 122, 127
- Jasp, 399

- Java threads, 250

- k-space, 1010, 1022
- Kalman filter, 965, 966, 970
 - augmented, 966
- Kaplan–Meier curves, 829
- Kaplan–Meier method, 769
- Karush–Kuhn–Tucker (KKT) condition, 860
- kernel
 - function, 851
 - kernel trick, 852
 - matrix, 854
 - mercer, 856
- kernel density, 1034, 1036
- kernel density estimation, 62, 307
- kernel estimation, 531, 967
- kernel smoother, 541
- keystroke timings, 1043, 1045
- knowledge discovery, 788
- Kolmogoroff metric, 659, 661, 668
- kriging, 498, 504, 510
- Kuiper metric, 668
- Kullback–Leibler discrepancy, 450

- lagged-Fibonacci generator, 45
- Lagrange multipliers, 567
- Laplace approximation, 458
- largest nonidentifiable outlier, 673
- Larmor frequency, 1003
- lasso, 639
 - computation, 640
- late binding, 420
- latent variables, 74
- Latin hypercube sampling, 512
- lattice, 43, 52, 57
- Law of Large Numbers, 730
- learning, 843
- least median of squares LMS, 684
- least squares, 623
 - computation, 623
 - explicit form, 623
 - Gauss–Markov theorem, 623
 - inference, 624
 - orthogonal transformations, 624
- least trimmed squares, 685, 686

-
- length of the shortest half, 669
 - Levenberg–Marquardt method, 647
 - leverage effect, 969
 - leverage point, 686
 - library, 393
 - likelihood, 954, 955, 964, 965, 972
 - function, 952, 962, 965, 967, 970, 972
 - intensity-based, 971
 - intractable, 720
 - marginal, 975
 - maximum, 954, 962, 972
 - simulated, 954, 959
 - likelihood ratio test
 - generalized linear model (GLM), 602
 - likelihood smoothing, 558
 - limited dependent variable, 952
 - linear congruential generator (LCG), 43, 49, 57
 - linear discriminant analysis, 815
 - linear feedback shift register (LFSR), 51, 53, 55, 57
 - linear recurrence, 42
 - linear recurrence modulo 2, 50
 - linear recurrence with carry, 49
 - linear reduction, 566
 - linear regression, 592, 622, 682
 - linear smoother, 541, 547
 - linear system
 - direct methods, 116
 - Gauss–Jordan elimination, 117
 - iterative refinement, 119
 - gradient methods, 124
 - conjugate gradient method, 125
 - Gauss–Seidel method, 125
 - steepest descent method, 125
 - iterative methods, 120
 - Gauss–Seidel method, 122
 - general principle, 120
 - Jacobi method, 122
 - successive overrelaxation (SOR) method, 123
 - link function, 559, 592, 594, 739
 - canonical, 595
 - linked brushing, 300
 - linked highlighting, 808
 - linked views, 300
 - linking, 388
 - local bootstrap, 486, 487
 - local likelihood, 559
 - local likelihood equations, 559
 - local linear estimate, 543, 544
 - local optimization, 180
 - local polynomial, 547, 559
 - local regression, 542, 616
 - local reversibility, 88
 - localized random search, 178, 182
 - location functional, 659, 662, 667, 669, 674, 678
 - location-scale-free transformation, 772
 - log-likelihood, 84
 - generalized linear model (GLM), 596
 - log-linear model, 614, 781
 - log-logistic distribution, 769
 - log-normal distribution, 769, 958, 971
 - log-rank statistic, 830
 - logistic distribution, 959
 - logit, 593
 - mixed, 959, 960, 971
 - mixed multinomial (MMNL), 953, 958–960
 - model, 959, 971
 - multinomial, 958
 - probability, 958, 959
 - logit model, 592, 596, 604
 - longitudinal, 98
 - longitudinal data, 614
 - loss function, 170, 844
 - low pass filter, 443
 - LR method, 129
 - LU decomposition, 106
 - M-estimation, 560
 - M-functional, 663, 664, 668, 669, 671, 675, 676, 682, 684
 - with a redescending ψ -function, 666
 - M-step (Maximization step), 139, 141
 - exponential family, 141, 148
 - factor analysis model, 158
 - failure-time data, 148
 - generalized EM (GEM) algorithm, 142

- normal mixtures, 146
- magnetic field inhomogeneities, 1011, 1014
- magnetic resonance, 1003
- magnetic resonance imaging, 1004
- magnetism, 1003
- magnetoencephalogram (MEG), 1006
- Mallow Cp, 438, 444, 450, 452, 453, 456
- MANET, 308–310
- margin, 850
- marginal distribution function, 773
- marginal likelihood, 72
- market risk, 912
- marketing, 953, 958
- Markov bootstrap, 483, 488, 489, 491
- Markov chain, 73, 75, 192
- Markov chain Monte Carlo (MCMC), 72, 162, 163, 458, 963, 966–969, 971–974
- Markov chain Monte Carlo (MCMC) algorithm, 720, 740
 - automated, 754
- Markov random field, 165
- Markov switching autoregressive model, 973
- masking effect, 671
- Mason Hypergraphics, 314
- Mathematica, 399
- mathematical programming, 181
- matrix decompositions, 104
 - Cholesky decomposition, 105
 - Givens rotations, 110
 - Gram–Schmidt orthogonalization, 112
 - Householder reflections, 109
 - LU decomposition, 106
 - QR decomposition, 108
 - SVD decomposition, 114
- matrix inversion, 115
- matrix linear recurrence, 50
- maximally equidistributed, 52, 55
- maximum full likelihood, 777
- maximum likelihood, 596, 597, 954, 962, 972
 - Monte Carlo (MCML), 962, 964, 965, 967–969, 971
 - quasi- (QML), 970
 - simulated, 954
- maximum likelihood estimate, 666, 770
- maximum likelihood estimation, 138
 - global maximum, 138, 143, 144
 - local maxima, 138, 143, 144, 150
- maximum partial likelihood, 777
- maximum score method, 714
- mean squared error, 184, 446, 523, 547
- measurement noise, 182
- median, 657, 659, 663, 668, 671, 674, 687, 689
- median absolute deviation MAD, 658, 663, 668, 671, 687
- median polish, 689
- menu hierarchy, 386
- Mersenne twister, 51, 54, 55, 58
- message, 407
- metaclass, 408
- metamodel, 499, 501, 502, 511
- method of composition, 97
- method of moments, 771
- Metropolis method, 72
- Metropolis–Hastings algorithm (MH), 163, 740, 741, 960
- Metropolis–Hastings method, 73
- micromaps, 319
- military conscripts, 743
- MIMD (multiple instruction stream–multiple data stream), 239
- MiniCAVE, 316
- minimum covariance determinant (MCD), 678
- minimum volume ellipsoid (MVE), 678
- mirror filter, 229
- misclassification cost, 818
- missing variables
 - simulation of, 743
- mixed model, 614
- mixed multinomial logit (MMNL), 953, 958–960
- mixing, 74
- mixing density/distribution, 958–960
- mixture
 - Poisson distributions, 727
- mixture models, 138, 150, 162
 - mixture of factor analyzers, 159, 164
 - normal mixtures, 145, 152, 160–162
- Mixture Sampler algorithm, 966
- mode

- attraction, 742
- mode tree, 531
- model
 - AR, 723, 728, 751
 - averaging, 728, 750
 - binomial, 726
 - choice, 747
 - generalised linear, 722
 - generalized linear, 739
 - index, 748
 - mixture, 744
 - probit, 736, 740
- model averaging, 728
- model choice, 726
 - and testing, 727
 - parameter space, 728
- model complexity, 444
- model domain, 405
- model selection, 438, 972, 973
 - generalized linear model (GLM), 602
- modified Bessel function, 933
- moment generating function, 204
- moment index, 570
- Mondrian, 308–310
- Monte Carlo, 172
 - confidence interval, 733
 - Markov chain (MCMC), 963, 966–969, 971–974
 - maximum likelihood (MCML), 962
 - with importance function, 731
- Monte Carlo EM, 154, 155, 163
- Monte Carlo maximum likelihood (MCML), 962, 964, 965, 967–969, 971
- Monte Carlo method, 37, 405, 498, 499, 730
 - and the curse of dimension, 736
- Monte Carlo techniques, 729
 - efficiency of, 734
 - population, 757
 - sequential, 757
- Moore’s law, 238
- mosaic map, 522
- mosaic plot, 307, 808
- mother wavelet, 221
- MPI (Message Passing Interface), 256
- multicollinearity, 625, 626
 - exact, 625, 626
 - near, 626
- multilevel model, 615
- multimodality, 974
- multinomial distribution, 975
- multinomial responses, 612
- multiple binary responses, 832
- multiple counting processes, 780
- multiple document interface, 387
- multiple failures, 779
- multiple recursive generator (MRG), 42
- multiple recursive matrix generator, 54
- multiple-block M–H algorithms, 73
- multiply-with-carry generator, 49
- multiresolution analysis (MRA), 217
- multiresolution kd-trees, 161
- multivariate smoothing, 557
- multivariate-t density, 85
- mutation, 188, 190

- Nadaraya–Watson estimate, 474, 541
- negative binomial distribution, 594
- nested models, 602
- network sensor, 1033
- neural network, 184, 799, 853
- New York Stock Exchange (NYSE), 970
- Newton’s method, 181, 646
- Newton–Raphson algorithm, 181, 186, 598, 959
- Newton–Raphson method, 560, 782
- Neyman–Pearson theory, 724
- no free lunch (NFL) theorems, 175
- node impurity, 817
- noisy measurement, 179, 180
- nominal logistic regression, 612
- non-nested models, 602
- nonhomogeneous Poisson process, 781, 783
- nonlinear least squares, 645
 - asymptotic normality, 645
 - inference, 647
- nonlinear regression, 622, 644
- nonparametric autoregressive bootstrap, 486
- nonparametric curve estimation, 471, 472, 491
- nonparametric density estimation, 518

- normal approximation, 733
- normal distribution, 658, 670
- normal equations, 543, 623
- normalization property, 219, 225
- normalizing constant, 731, 737
 - ratios of, 737
- novelty detection, 843, 868
- NOW (network of workstations), 241
- nuisance parameter, 781, 783
- null deviance, 604
- NUMA (non-uniform memory access), 240
- numerical standard error, 78
- nViZn, 321
- Nyquist ghosts, 1011, 1014

- object, 406
- object composition, 410, 418
- Object oriented programming (OOP), 405
- object, starting, 432
- Occam's razor, 444, 457
- offset, 601
- Old Faithful geyser data, 519, 520
- one-way analysis of variance, 686
- one-way table, 686
- OpenMP, 251
- optimization/optimizer, 959, 964, 967, 972
- order of menu items, 386
- ordered probit model, 613
- ordinal logistic regression, 613
- ordinary least squares (OLS), 963, 964
- orthogonal series, 546
- orthogonality property, 219, 225
- outlier, 657, 665, 670, 671, 680, 684, 689, 969
- outlier detection, 843, 868
- outlier identification, 681, 687
- outlier region, 672
- outwards testing, 673
- overdispersion, 612
- overfitting, 799
- oversmoothing, 524, 528

- panel data, 615, 952, 975
- panning, 301
- parallel computing, 238
- parallel coordinate display, 303
- parallel coordinate plot, 303, 366
- parallel coordinates, 806, 1039, 1042
- parameter
 - of interest, 723
- parameter space
 - constrained, 720
- parameter-expanded EM (PX-EM) algorithm, 160
- Parseval formula, 211
- partial autocorrelation, 724
- partial least squares, 641
 - algorithm, 642
 - extensions, 643
 - latent variables, 642
 - modified Wold's R, 642
 - nonlinear regression, 649
 - Wold's R, 642
- partial likelihood, 775, 777
- partially linear models, 704
- particle systems, 757
- password cracking, 1043
- pattern recognition, 184
- Pearson statistic, 601
- penalized least squares, 441, 544
- penalized likelihood, 138, 160, 165, 559
- perfect sampling method, 99
- periodogram, 207
- permutation tests, 482
- physiological noise, 1012, 1017
- pie chart, 307
- piecewise polynomial, 545
- pilot estimate, 554
- pixel grand tour, 305
- plug-in, 468–470
- plug-in bandwidth selection, 553
- PMC vs. MCMC, 761
- point process, 971
- Poisson data, 593
- Poisson distribution, 57, 779, 972
- Poisson process, 63
- poly- t distribution, 722
- polymorphic class, 420
- polymorphism, 419
- polynomial lattice, 52
- polynomial LCG, 53

-
- polynomial regression, 499, 508, 513
 - polynomial terms, 606
 - population, 187
 - population Monte Carlo (PMC) techniques, 757
 - positron emission tomography (PET), 1006
 - posterior density, 83, 955, 959, 960, 962, 967, 968, 974
 - posterior distribution, 722
 - posterior mean, 955–957, 967–969, 974
 - posterior probability, 146, 150, 161
 - power expansions, 778
 - power method, 127
 - power parameter, 772
 - prediction, 551, 723
 - sequential, 724
 - predictive modeling, 794
 - predictive squared error, 446
 - PRESS, 454
 - primitive polynomial, 43, 50
 - principal components analysis (PCA), 566, 632
 - principal components regression, 632, 633
 - choice of principle components, 633
 - principal curve, 582
 - prior
 - proper, 730
 - prior (density), 955–957, 960, 966, 967, 969, 972, 974
 - informative, 974
 - uninformative, 956, 967
 - prior distribution, 720
 - conjugate, 722
 - selection of a , 721
 - prior-posterior summary, 84
 - probability of move, 73
 - probit
 - model, 958, 972
 - multinomial, 958
 - multinomial multiperiod, 952–954
 - multivariate, 953, 958
 - static multinomial, 955
 - probit model, 596, 605
 - probit regression, 93
 - problem domain, 405
 - process control, 185
 - process forking, 245
 - productivity, 390
 - program execution profiling, 1048
 - progress bar, 390
 - projection, 302
 - projection index, 569
 - projection pursuit, 305, 557, 568
 - projection pursuit guided tour, 305
 - projection pursuit index, 305
 - projection step, 583
 - proportion, 568
 - proportional hazard, 830
 - proportional hazards model, 710, 776
 - proposal
 - adaptive, 761
 - multiscale, 759
 - proposal distribution, 73
 - prosection matrix, 302
 - prosections, 302
 - proximity, 825
 - pruning, 799
 - pseudo data, 468
 - pseudo-likelihood, 612
 - pseudorandom number generator, 37
 - Pthread library, 248
 - pulse sequence, 1008
 - PVM (Parallel Virtual Machine), 253
 - QR decomposition, 108
 - QR method, 128
 - quadratic principal components analysis (QP-CA), 572
 - quality improvement, 184
 - quasi-likelihood, 612
 - quasi-maximum likelihood (QML), 970
 - queuing systems, 184
 - R, 399, 763
 - radial basis networks, 853
 - random effects, 98, 154, 155
 - random forests, 825
 - random graph, 1046
 - random noise, 172
 - random number generator, 37, 410

- approximate factoring, 46
- combined generators, 47, 54, 56, 58
- definition, 38
- figure of merit, 44, 51
- floating-point implementation, 46
- implementation, 46, 56
- jumping ahead, 39, 48, 51
- non-uniform, 58
- nonlinear, 55
- period length, 39, 50, 54
- physical device, 38
- power-of-two modulus, 47
- powers-of-two-decomposition, 46
- purely periodic, 39
- quality criteria, 39, 59
- seed, 38
- state, 38
- statistical tests, 41, 56
- streams and substreams, 39, 58
- random numbers, 410
 - common, 959, 964
 - pseudo-, 959
 - quasi-, 959
- random permutation, 473, 474, 477, 482
- random permutation sampler, 973
- random perturbation vector, 184
- random search, 176
- random walk M–H, 80
- Rao–Blackwellization, 95
- rate of convergence, 144, 152, 156, 159–161
 - rate matrix, 144
- ratio
 - and normalizing constants, 731
 - importance sampling for, 737
 - of integrals, 731
 - of posterior probabilities, 725
- ratio-of-uniforms method, 63
- real-number coding, 190
- recursive partitioning, 366
- recursive sample mean, 1034
- red-green blindness, 392
- redescending ψ -function, 665
- reduced conditional ordinates, 96
- reformatting, 302
- REGARD, 308, 309, 318
- regression depth, 685
- regression equivariant, 683
- regression functional, 682
- regression splines, 545
- regression trees, 835
- regression-type bootstrap, 486, 487
- regressor-outlier, 686
- regularization, 846
- rejection method, 62, 64
- rejection sampling, 967
- relative projection Pursuit, 571
- remote sensing data, 521
- resampling, 468, 469, 471–476, 478, 481–483,
485–492, 556
- resampling tests, 476, 478
- rescaling, 302
- residual, 688
- residual sum of squares, 598
- residuals
 - generalized linear model (GLM), 601
- resistant one-step identifier, 686
- resolution of identity, 217
- response-outlier, 686
- restricted maximum likelihood, 460
- reverse move,
 - probability, 749
- reversible, 76
- reversible jump MCMC, 748, 752
- ridge regression, 340, 635
 - almost unbiased, 637
 - almost unbiased feasible, 637
 - bias, 635
 - choice of ridge parameter, 635–637
 - feasible generalized, 637
 - generalized, 636
 - minimization formulation, 636
 - nonlinear regression, 648
 - reduced-rank data, 637
- risk, 844
 - empirical, 844
 - expected, 845
 - regularized, 846
 - structural minimization, 847
- risk measure, 912
- Robbin–Monro algorithm, 757

-
- robust, 683
 - robust functional, 671
 - robust location functional, 661
 - robust regression, 560, 682, 684
 - robust scatter functional, 680
 - robust statistic, 657, 659, 661
 - robustness, 659, 663, 668
 - root node, 817
 - root-finding, 171, 182
 - rotation, 302
 - roulette wheel selection, 189

 - S-functional, 666, 679, 681, 685, 686
 - S-Plus, 399
 - sample mean, 771
 - sampler performance, 74
 - SAS, 399
 - Satterwaite approximation, 555
 - saturated model, 598
 - saturation brushing, 307
 - scalable EM algorithm, 161
 - scale functional, 663, 669
 - scales, 353
 - scaling algorithm, 742
 - scaling equation, 218
 - scaling function, 217
 - scanner data, 955
 - scatter diagram, 519, 521
 - scatterplot, 298, 387
 - scatterplot matrix, 298, 391
 - schema theory, 192
 - search direction, 172
 - secondary data analysis, 790
 - selection, 188, 189
 - selection sequences, 301
 - self-consistency, 143
 - semiparametric models, 557, 700
 - semiparametric regression, 615
 - sensitivity analysis, 498
 - sensor placement, 185
 - serial test, 57
 - setter, 406
 - shape parameter, 770
 - shared memory, 239
 - shortcut, 386

 - shortest half, 661, 666, 678, 684
 - shrinkage estimation, 634
 - shrinking neighbourhood, 660
 - sieve bootstrap, 485, 486, 490, 491
 - SIMD (single instruction stream–multiple data stream), 239
 - simulated maximum likelihood (SML), 954, 959–961, 965, 967, 968
 - quasi-random, 959
 - simulated moments, 954, 959
 - simulated scores, 959
 - simulated tempering, 98
 - simulation, 498, 499, 504, 509, 513, 729, 952, 954, 960, 962, 965, 966, 970, 975
 - simulation-based optimization, 173, 184
 - simultaneous perturbation SA (SPSA), 183
 - single index model, 615, 701
 - single trial fMRI, 1009
 - SISD (single instruction stream–single data stream), 239
 - slammer worm, 1048
 - slash distribution, 670
 - slice sampling, 92
 - sliced inverse regression, 584
 - slicing, 302
 - smooth bootstrap, 759
 - smoothed maximum score, 714
 - smoothing, 540
 - smoothing parameter, 442, 453, 459, 460, 523, 547
 - SMP (symmetric multiprocessor), 240
 - soft thresholding, 443
 - software reliability, 779
 - sparse matrices, 129
 - sparsity, 860
 - specification search, 699
 - spectral density, 207
 - spectral test, 44
 - spectrogram, 209
 - speech recognition, 814
 - Spider, 309
 - SPIEM algorithm, 161
 - spine plot, 307
 - spline, 438, 441, 447, 455, 456
 - spline smoother, 544, 545

- spreadplots, 307
- SPSA Web site, 184
- SPSS, 399
- SQL, 340
- squeeze function, 63
- SRM, *see* structural risk minimization
- stably bounded algebraic curve, 579
- standard deviation, 658, 663, 668, 671
- standard errors, 151–153
- starting (initial) value, 143, 144, 150, 151
- state space, 964, 966
- state space model
 - Gaussian linear, 965, 970
- stationarity, 724, 961, 974
- statistical computing, 5
- statistical functional, 659
- Statistical Parametric Mapping (SPM), 1022
- steepest descent, 180
- steepest descent method, 125
- Stein-rule estimator, 634
- stereoscopic display, 313
- stereoscopic graphic, 313
- stochastic approximation, 180
- stochastic conditional duration (SCD) model, 970, 971
- stochastic gradient, 180
- Stochastic optimization, 170
- stock trading system, 970
- stopping rule, 757
- streaming data, 1033
- structural risk minimization (SRM), 847, 849
- structure parameter, 783
- Structured Query Language, 340
- Student-*t* distribution, 968, 969
- subsampling, 472, 482–484
- subtract-with-borrow, 49
- successive overrelaxation (SOR) method, 123
- supervised learning, 791
- supplemented EM (SEM) algorithm, 151, 154
- support, 801
- support vector machine, 843
 - decomposition, 861
 - linear, 849
 - optimization, 857
 - sequential minimal optimization (SMO), 863
 - sparsity, 860
- support vector novelty detection, 868
- support vector regression, 867
- surrogate data tests, 483
- surrogate splits, 824
- survival analysis, 147
- survival function, 768
- survival model, 560, 614
- survival rate
 - variance, 760
- survival trees, 829
- susceptibility artifacts, 1013
- SV model, 952, 961, 971
 - canonical, 961, 962, 964, 965, 967, 968, 970
 - multivariate, 969
- SVD decomposition, 114
- symmetric command sequence, 393
- syn cookie, 1033
- SYSTAT, 399
- systems of linear equations
 - direct methods, 116
 - Gauss–Jordan elimination, 117
 - iterative refinement, 119
 - gradient methods, 124
 - conjugate gradient method, 125
 - Gauss–Seidel method, 125
 - steepest descent method, 125
 - iterative methods, 120
 - Gauss–Seidel method, 122
 - general principle, 120
 - Jacobi method, 122
 - SOR method, 123
- Table Production Language (TPL), 352
- tailored M–H, 81
- tailoring, 88
- TAQ database, 970
- target tracking, 173
- Tausworthe generator, 51, 53
- Taylor series, 548
- Taylor series expansions, 782
- TCP three-way handshake, 1032
- t*-distribution, folded, 735

-
- tempering, 54
 - terminal nodes, 819
 - termination criterion, 192
 - Tesla, 1003
 - thinning, 63
 - threading, 247
 - threshold parameters, 770
 - thresholding, 213
 - time series, 470, 472–474, 480, 482–492, 952, 961, 974, 975
 - tissue contrast, 1005
 - tournament selection, 189
 - traffic management, 185
 - training data, 791
 - transform
 - continuous wavelet, 215
 - discrete Fourier, 206
 - discrete wavelet, 226
 - empirical Fourier–Stieltjes, 204
 - fast Fourier, 226
 - Fourier–Stieltjes, 203
 - Hilbert, 209
 - integral Fourier, 208
 - Laplace, 204
 - short time Fourier, 209
 - Wigner–Ville, 210
 - windowed Fourier, 209
 - transformation
 - Box–Cox, 203
 - Fisher z , 201
 - transformation models, 706
 - transformed density rejection, 63
 - transition kernel, 75
 - transition matrix, 193
 - translation equivariant functional, 677
 - transmission control protocol, 1031
 - transparent α -level contour, 307
 - trapping state, 745
 - tree growing, 817
 - tree pruning, 818
 - tree repairing, 833
 - Trellis display, 391
 - triangular distribution, 958
 - trigonometric regression, 438, 440, 447, 455, 456
 - trojan program, 1035, 1036
 - Tukey’s biweight function, 665, 681
 - twisted generalized feedback shift register (GFSR), 51, 54, 55
 - two-way analysis of variance, 688
 - UMA (uniform memory access), 240
 - unbiased risk, 450, 453, 460
 - unbiased risk estimation, 553
 - under-fitting, 845
 - Unified Modelling Language (UML), 406, 411
 - uniform distribution, 37, 958, 974
 - uniformity measure, 40, 51
 - unit measurement, 355
 - unobserved (or latent) variables, 952
 - unobserved heterogeneity, 710
 - unpredictability, 42
 - unsupervised learning, 791
 - user profiling, 1043
 - utility/utilities, 953–955, 958, 959, 961
 - validation data, 791
 - Value at Risk (VaR), 912, 914
 - copulas, 942
 - vanGogh, 399
 - vanishing moments, 225
 - Vapnik–Cervonenkis class, 674
 - variable
 - auxiliary, 747
 - variable selection, 438, 627
 - all-subsets regression, 629
 - branch and bound, 630
 - genetic algorithms, 630
 - backward elimination, 627
 - cross validation, 630
 - forward selection, 629
 - least angle regression, 629
 - stepwise regression, 627
 - variance estimation, 550
 - variance reduction, 59
 - variance reduction technique, 500
 - varset, 341, 342
 - VC-bound, 848
 - VC-dimension, 847
 - VC-theory, 847

- vector error-correction model, 955
- Virtual Data Visualizer, 317
- virtual desktop, 389
- virtual reality (VR), 295, 313, 314, 316, 317, 322
- Virtual Reality Modeling Language (VRML), 317
- visual data mining (VDM), 295
- volatility of asset returns, 952, 961
- voting, 952, 958
- VoxBo, 1022
- VRGobi, 314–316

- W*-transformation, 772
- Wasserstein metrics, 829
- waterfall plot, 1037
- wavelet domain, 226
- wavelet regularization, 846
- wavelets, 212
 - Daubechies, 224
 - Haar, 223
 - Mexican hat, 215
 - periodized, 232
 - sombrero, 215
- Weibull density function, 770
- Weibull distribution, 769, 770, 970, 971
- Weibull process model, 782
- weight function, 541
- weights
 - generalized linear model (GLM), 601, 611
- wild bootstrap, 486, 487, 491
- winBUGS, 720, 763
- window titles, 1043
- Wishart distribution, 957
- working correlation, 615

- XGobi, 305, 309, 311–318
- XML, 340
- XploRe, 399, 915, 920, 935

- zooming, 301