Charles DiMaggio

SAS for Epidemiologists

Applications and Methods

Springer

SAS for Epidemiologists

Charles DiMaggio

SAS for Epidemiologists

Applications and Methods

Springer

Charles DiMaggio, PhD
Departments of Anesthesiology
and Epidemiology
College of Physicians and Surgeons
Mailman School of Public Health
Columbia University
New York, USA

ISBN 978-1-4614-4853-2 ISBN 978-1-4614-4854-9 (eBook)
DOI 10.1007/978-1-4614-4854-9
Springer New York Heidelberg Dordrecht London

Library of Congress Control Number: 2012947994

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

For Liz, Katie, and CJ. Thank you for being there when I come home from epidemiology.

Foreword

Advances in information technology have transformed the field of epidemiology in profound ways. Data, which had been the most valuable commodity of the empirical sciences, are increasingly available and accessible. The quantity of data rises exponentially on a daily basis. The forms and sources of data are also multiplying. These changes are driving epidemiology into the era of "big data science" and converting more and more epidemiologists into data analysts. As a result, practical skills to manage large and complex data sets and make sound use of them are of increasing importance to future epidemiologists. This book by Dr. Charles DiMaggio is a valuable addition to the toolbox for epidemiology students, public health professionals, and researchers in the health-care industry.

This book distinguishes itself from other applied SAS texts in three notable aspects. First, it is extremely reader-friendly. Dr. DiMaggio has taught the introductory SAS course at Columbia to hundreds of students for over a decade. This book is based primarily on his lectures and is written in a conversational style. The materials are presented in a way that is easy to understand and interesting to learn. In addition to the technical know-how, each chapter contains some "clinical pearls" – insight and wisdom that are unavailable in other applied SAS books. In fact, the reader may feel more like being engaged in a conversation with Dr. DiMaggio than studying a monotonic, mind-numbing how-to manual. Second, this book focuses on the fundamentals. In the first five chapters, Dr. DiMaggio explains in painstaking detail the most basic functions of data input, management, and exploration. These chapters are followed by intermediate statistical analyses of epidemiologic data, both categorical and continuous. Going through these chapters will not make the reader an expert SAS programmer, but it will provide the reader with the necessary skills to perform analytical responsibilities required for a master's level epidemiologist. Finally, this book is filled with illuminating examples from actual epidemiology research projects. It is written for aspiring epidemiologists by an experienced epidemiologist. These examples are used not only for procedural demonstrations but also for explanations of important epidemiological concepts such as confounding, disease odds ratio, and exposure odds ratio.

As one of the most sophisticated statistical packages, SAS is so kaleidoscopic that first-time users often find it intimidating. Public health students will find this comprehensive text especially helpful for overcoming their initial fears and confusion. It can serve as a textbook for introductory courses on SAS applications as well as a self-study reference for epidemiologists and other health professionals.

New York, NY, USA Guohua Li
Columbia University M. Finster

Preface

The path to this book began, as these things so often do, when I was asked to teach a course. In this case, a semester-long class for master's students on epidemiologic analyses using SAS. Over a few years of preparing for and teaching the material I confronted a combination of practical and conceptual considerations that led me to believe that perhaps there was room for another book about SAS.

On a practical level, working with a program like SAS is a skill I consider necessary for all graduating master's epidemiologists. To be honest, the necessity that the program actually be SAS is based on a circular argument. Many employers of epidemiologists use SAS because their current analysts use SAS, and newly minted analysts will compel additional future analysts to use SAS. This reliance on SAS of potential employers of master's-level epidemiology students may change in the future, but my sense is that it will not be anytime soon. While the practical motivation to learn SAS is somewhat self-fulfilling, it does not detract from the capabilities that made SAS an important skill in the first place. And, does it make the choice of SAS any less necessary. As I sit and write this, a quick search on the *New York Times* jobs link returns 15 epidemiology jobs in the New York City area. A search for SAS returns 457 hits. When I do this search on the first day of class, with generally the same results, there is invariably an increased interest among the students in spending a few hours a week learning SAS.

The kinds of SAS-related work that master's-level epidemiologists are called upon to undertake do not exceed some fairly straightforward categorical and continuous data analyses. There was, though, no book that addressed this material in a similarly straightforward fashion. The feedback I've received from the past students is that the procedures covered in this material account for a good majority of their daily activity and that knowing how to do those things helps set the stage for learning more advanced material.

On a conceptual level, the role of statistical software in epidemiologic practice is in a state of flux, and the kinds of data and analyses epidemiologists are being called upon to work with are evolving into what might be called the era of "big data" and the rise of "computational epidemiology." SAS is tailor-made to deal with the kinds of huge data sets that are becoming routine in epidemiology. That there has been

an explosion in the availability of administrative and routinely collected health data, free and open-source data, social media data, and other online data is clear. That the data are amenable to reliable or valid analyses is less clear. The basics, about missing and incorrect values, about confounding, about bias, about study design, are if anything even more important. The data can inform, but we may have to teach them to speak clearly, and in a language that is epidemiologically valid.

Fortunately, SAS is more than up to the task. It has a facility for dealing with extremely large data sets that I have found unsurpassed in other statistical programs. SAS allows epidemiologists to pay special attention to the necessary (though not glamorous) initial steps of reading in, preparing, and cleaning large amounts of data, when early errors or missteps will be amplified throughout the analysis, sometimes in ways that are difficult to trace to their origins. For this reason, fully the first third of this book addresses using SAS to read in and manipulate data to get them into a form that makes epidemiologic sense.

The one aspect of preparing and teaching this material that I did not expect was that it was actually fun. I'm certain this says more about me than it does about the material. But perhaps a kind of geeky enjoyment of some of the practical aspects of epidemiologic methods, like learning how to use SAS, is a sign that you've chosen the right profession. I tried to capture some of what I found interesting and enjoyable by using examples and materials that have practical relevance to epidemiologic practice.

I have come to appreciate that public health practice requires a long-term view, and that you may not always (or even frequently) see the effects of your work. The effects of teaching public health are even farther removed from immediacy. Despite the practical aspects underlying this book, the ultimate motivation is as ephemeral as public health practice itself. In the end, I hope to contribute in some small way to the efforts of someone I haven't met, to improving the health, happiness, and well-being of someone who may not even be born yet.

New York, NY, USA Charles DiMaggio

Acknowledgments

I was fortunate enough to learn (and hopefully pass on to you) these concepts and procedures from a small but uniformly excellent set of books and the scientists who wrote them [1–12]. I try to cite areas where they were particularly illustrative, but I borrowed from them shamelessly throughout.

I am especially indebted to the SAS Institute for allowing me to draw on their training manuals, course notes, and data sets. If you want a thorough grounding in SAS, I strongly recommend you take one of their outstanding training classes. I also appreciate the willingness of the New York State Department of Health to allow me to use a version of their Statewide Planning and Research Cooperative System (SPARCS) data that figure so prominently in these pages.

Finally, many thanks to my colleagues in the departments of anesthesiology and epidemiology at Columbia University in New York, for their support and feedback, and to my stellar teaching assistants, Joanne Brady, and George Loo, who know SAS so much better than I do, and who helped breathe life into this material.

Contents

Part II Descriptive and Categorical Analysis

Chapter 1
Introduction

Abstract The primary objective of this book is to provide you with the tools to use the SAS statistical software package in the practical conduct of basic epidemiological research analyses. To reach that objective, you will need to first become familiar with the SAS computing environment so you can create, open, and retrieve files and learn how to read, clean, and manipulate data. You will then become familiar with conducting descriptive analyses in SAS and understand how to conduct some slightly more advanced analytic tasks like analysis of variance (ANOVA), linear regression, and categorical data analyses.

1.1 About SAS

Since you're reading this, you probably have some sense of what SAS is. But it never hurts to start from first principles. SAS stands for Statistical Analysis System. It is a powerful suite of procedures, and the software language to manipulate them, that is well known, well documented, and flexible enough to handle most any statistical task. It is, essentially, an industry standard, with ninety-six of the top 100 Fortune 500 companies using it. It is also broadly utilized by government public health agencies, both federal, like the Centers for Disease Control and Prevention (CDC) and the National Institutes of Health (NIH), and local like the New York City Department of Health and Mental Hygiene.

A number of factors contribute to SAS's popularity. It works in a desktop Windows environment but can be deployed as a command-line, enterprise-level Unix system. It can import almost any type of data (ascii, delimited, Excel, etc.) It can be manipulated to handle almost any statistical procedure. It is particularly powerful at handling the extremely large data sets that are increasingly available to public health practitioners and that constitute the databases for many surveillance activities. Some agencies, such as the National Center for Health Statistics, provide downloadable data in SAS format along with SAS syntax to read and interpret it. This in itself is a strong incentive for epidemiologists to consider learning SAS.

C. DiMaggio, *SAS for Epidemiologists: Applications and Methods*,
DOI 10.1007/978-1-4614-4854-9_1,

1.1.1 Alternatives to SAS

Perhaps the principle disadvantage of using SAS is the yearly licensing fee which may be prohibitively expensive for students and some researchers, particularly those without the benefit of institutional affiliations to help offset the cost.[1] There are other statistical programs that are (certainly) cheaper and (perhaps) easier to use. Chances are, in fact, that if you spend any amount of time doing epidemiologic analysis, you will be called on to use more than one program anyway. Here's a selective and highly opinionated roundup of some of the usual suspects:

- Excel. This ubiquitous spreadsheet program has a quick learning curve, many useful statistical functions, and the advantage of being readily understood and shared for collaborative purposes. Many public health and epidemiological researchers will find, though, that they soon outgrow it when they develop a need for more advanced and specialized procedures, data manipulation, and graphics.
- SPSS. Nice if you like graphical user interfaces and a menu-driven approach to data analysis. Depending on the type of work you do, this may very well be the only statistical software package you will need. It is particularly well suited to procedures such as factor analysis and is very popular among social scientists. (In fact, SPSS stands for Statistical Package for Social Science.) The spreadsheet-like interface provides a relatively painless transition from a program like Excel. An added advantage for many users is that the package can be purchased during one's graduate education obviating the need for costly yearly licensing fees.
- R. A popular open-source freeware alternative to proprietary statistical software packages. The language is similar to S-Plus. As more researchers and statisticians contribute to its development, and as documentation proliferates, it will be an increasingly viable option for researchers and epidemiologists. There are few, if any, procedures it cannot handle. The graphics are as good as or (in many cases) better than those found in expensive packages like SAS. It even includes mapping and spatial analytic tools to rival those found in packages costing thousands of dollars. It has a lot to recommend it, and I'm a big fan. I do, though, find it less flexible in handling the large public health data sets SAS handles with ease.
- EpiInfo. Developed by the CDC for their own investigators, this program has something for everyone; it even comes with a nice mapping module. Epidemiologists will appreciate the sample size calculators, the simple intuitive 2×2 table calculators, and questionnaire development and analysis programs for case-control studies. Every practicing medical epidemiologist owes it to themselves to download this free program.
- SUDAAN. This software program was developed expressly for the purpose of analyzing survey data. The procedures themselves are familiar, but the analyses take account of complex sampling strategies, nesting of variables within strata, and correlations among data elements. Available in either a stand-alone version

[1] There is a reason why you tend to find SAS at large companies and government agencies.

or as a SAS-callable version, the program is utilized for the analysis of many large, ongoing US government health surveys such as CDC's Behavioral Risk Factor Surveillance System. Recently, SAS introduced survey procedures that may obviate the need for an add on like SUDAAN.

1.1.2 Why SAS, Then?

Simply put, it is the most highly rated, most comprehensive, most robust, most reliable, and best documented statistical program out there. If you are a master's level epidemiologist or public health professional, knowing SAS will enhance your ability to get a job and allow you to work more effectively once you get that job.[2]

1.2 About This Book

This is a book about SAS for epidemiologists (or other public health researchers) insomuch as I am an epidemiologist who uses SAS for my work. There are folks who have devoted their careers to writing and developing SAS programs. There are other folks who spend their time doing nothing but work in SAS. I am not one of them. I know that there are other, perhaps more elegant, ways of accomplishing much of what I will present. There is an entire universe of SAS out there. But, I have worked with SAS for a decade and have carved out a small area of that universe that has been useful and productive for the conduct of my research. This book is an attempt to make that area of SAS accessible to epidemiologists, public health workers, researchers, and other health professionals whose day-to-day activity calls on them to analyze data.

This material was developed for a one-semester component of a master's level program in epidemiology. I don't assume you know anything about SAS. If you do, you may find some of the early going tedious. I do assume, though, that you already have some knowledge of basic epidemiology, such as study designs like case-control, cohort, and randomized clinical trials, and know what a confounder, a mediator, and an interaction term are. I also assume you know some introductory master-level biostatistics. I will accept that you may have forgotten or become a bit fuzzy about what you learned, and I'll briefly review some important statistical concepts as they are introduced. This is not, though, a primer or textbook on biostatistics of which many excellent examples are available. Optimally, as you are reintroduced to basic epidemiological and statistical concepts, and apply them using SAS, you will not only learn something about SAS, but also gain a better understanding of epidemiology and statistical analysis as well.

[2] A Few years back, the master's students at my university essentially demanded that we teach them SAS so they could better compete for jobs.

We will be using the SAS programming language to conduct analyses. There is a menu-driven graphical user interface version of SAS available, but most researchers and epidemiologic analysts use the syntax-driven SAS-language-based approach. Writing syntax presents a bit more of a learning curve but is amply rewarded by the increased control, flexibility. and ability to document and recall analyses. It also gently introduces some programming skills, which epidemiologists are increasingly called on to have.

SAS is, for all intents and purposes, an exclusively Windows environment. There is a command-line, Unix version of SAS, but I can't imagine most epidemiologists will encounter it. You can, with some difficulty, port SAS to a Linux environment, but not (at least to my experience) with full functionality. The last time SAS produced a product for Mac users was version 6.2, so a native Mac version is really not an option.[3] So all the directions and screen grabs in this book are pictures of Windows.

1.2.1 Goals

My goals for readers of this book are that by the end, you should be able to:

- Read raw data into SAS
- Manipulate and clean data sets in SAS through printing, sorting, merging and the use of conditional expressions
- Apply simple statistical and graphical procedures for the descriptive analysis of data
- Conduct simple and stratified categorical data analyses in SAS and understand the concept of confounding within that context
- Conduct ANOVA in SAS, interpret SAS ANOVA output, and be aware of the concept of statistical interaction within that context
- Conduct correlation and linear regression in SAS. Interpret SAS output for correlation and linear regression
- Understand and apply the concept of residual analysis for regression diagnostics.
- Understand the concepts of model building within the context of linear regression

1.2.2 How to Use the Book

The material in this book covers a master's level semester-long introductory course on applied epidemiological analysis in SAS. The data sets that I use to illustrate concepts and procedures, and which form the basis for the problems at the end of each chapter, are available to instructors who contact me through the publisher.

[3] I recently switched from Linux to Mac. I run SAS (very nicely) under Windows XP using a Parallels virtual machine.

This is in keeping with my agreement with the folks who have been gracious enough to allow me to use their data. This doesn't preclude using the book for self-study, but SAS is a language, and the only way to learn a language is to use it.[4]

1.2.2.1 About Those Grey Boxes and Their Hyperlinks…

Throughout the text, I sometimes set off additional information in grey boxes. If you are using an electronic version of the book, the titles of those little snippets of info are song titles that link to online videos.

EVERYDAY I WRITE THE BOOK

Something like this.

The songs themselves involve in some way some aspect of the material in the box, and usually reflect my (heavily 1970s and 1980s influenced) musical tastes. Sometimes they're just the closest titles I could find that include *something* about the material in the box.[5] Consider them an opportunity to take a little break and refill your coffee mug.

[4] I may, in the future, come up with a way to get the data to individuals, so it doesn't hurt to contact me.

[5] You try finding a popular song about Macros.

Part I
Working with Data in SAS

This first section of material is, in many respects, the most important. SAS is most usefully a tool for the secondary analysis of large, existing data sets. But first, you must get the data into SAS. Then, the data have to be examined, and invariably manipulated, to draw valid and reliable epidemiological inferences. SAS provides arguably some of the best solutions available to epidemiologists for this process.

Chapter 2
The SAS Environment

Abstract In this chapter, we begin to become familiar with the basic SAS working environment. We introduce the basic 3-screen layout, how to navigate the SAS Explorer window, the two basic types of SAS programs, the kinds of data with which you will work in SAS, and how to get help.

2.1 The SAS Screen

When you start up SAS, your screen will typically be split up into three frames (Fig. 2.1).

On your left you can tab (at the bottom) between the "SAS Explorer", to navigate to files and folders, and SAS "Results" which allows you to navigate to the different statistical analyses and graphs you create during a SAS session.

On your right, you will see the Editor window on the bottom half of the screen where you will write and submit your programs and commands and the Log Window on the top portion of the screen, where SAS will give you messages (including flagging errors) about your submitted programs. The Output Window lurks behind the Editor and Log. When you submit syntax to perform analyses, it instantly comes to the fore to display your results. You can, though, bring it up to the front at any time by left clicking on Output on the bottom of your screen.

You left click on the frame of a window or the tab to make a window active. Your toolbar and available commands are context specific, i.e., they change depending on which window is currently active.

2.2 The Program Editor

The Editor window comes in two flavors. You can use the program editor window or the enhanced program editor window.

C. DiMaggio, *SAS for Epidemiologists: Applications and Methods*,
DOI 10.1007/978-1-4614-4854-9_2,

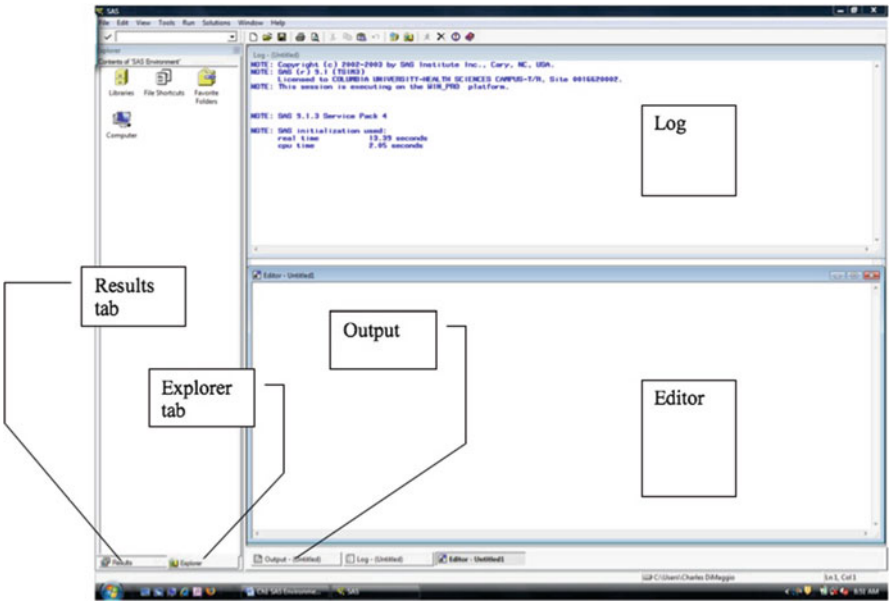


Fig. 2.1 The SAS screen

The program editor window is available across all computing platforms. Only one program editor window can be open at a time, and when you submit a command, it disappears and needs to be brought back up with the "recall" command. Of course since you don't have an editor window anymore, it's not easy to submit a command. You will find it again under the View menu at the top of your screen.

The enhanced editor window is available in MS Windows environments, where it is the default and just appears as Editor'. It has some nice features, not the least of which is that it doesn't disappear when you submit a command. It also color codes your syntax. Data and procedure keywords appear in blue and variables in black. Helpfully, it also flags errors in syntax by coloring them red. It's the only way to go.

WINDOWS.

If you accidentally close down a window, you can bring it back up by going to the View menu at the top of your screen and clicking on the window you want.

2.3 SAS Statements

You write SAS commands or statements in the Editor window. A line of SAS syntax almost always *starts with a keyword* of some kind, usually **DATA** or **PROC**, (we'll talk about those soon) and *ends with a semicolon*. You can have any number of SAS statements each requesting that SAS do something different. In a very common sense way, SAS will run statements when it sees the word RUN.

Once you've written your commands, make sure *each statement ends in a semicolon* and that all the statements you want to run are followed by a RUN command and that the RUN command also *ends in a semicolon*.[1] You can then submit your syntax. In MS Windows, you submit your statements by (1) highlighting it by left click-dragging your mouse across the syntax and then (2) clicking the little running-man icon on the tool bar. SAS then processes the syntax and then returns your results in the Output window as well as some information, such as how long the process took, if SAS encountered any errors, etc., in the Log window.

RUNNING

If you don't highlight the syntax you want run and just click the running-man icon, SAS will run every command in the Editor window. If you have been working on a lot of code, creating and manipulating files, you may end up destroying a lot of your work.

2.4 Comments

SAS lets you write little notes to yourself or to others who may try to read or run your program. There are two ways to indicate to SAS that what you are typing is not a command and that it should ignore it. You can either enclose the comment with a slash–asterisk asterisk–slash, like so:

```
/* comment */
```

or you can enclose it with an asterisk and a semicolon, like so:

```
* comment ;
```

The second approach is useful if you want to insert a short comment off to the side of a line of syntax or if you want to convert a section of code to text (just insert an * at the beginning of the line of syntax). Then the next time you want to run that syntax, just remove the asterisk.

[1] You may be beginning to detect the importance of semicolons in SAS.

I suggest you use comments liberally. They not only help analysts coming after you, but also they help you when you forget what exactly you were trying to accomplish with a particularly gnarly line of syntax.

2.5 Quick Demonstration of an SAS Program

So what does a SAS program look like? Something like this (Fig. 2.2).

Here you see the command syntax written in the Editor window. I've enlarged the default font size so you can read it. Don't worry about what the data represents at this point. This is just a brief, initial, demonstration.

In the Editor window, notice a couple of things. First, look at all those semicolons. Next, notice that the code is color coded. The procedure I'm requesting ("proc univariate") is dark blue, as is the final "run" statement. The subcommands under that procedure, such as specifying the data set to work on ("var") the types of graphs I want for that variable ("histogram" and "probplot") are all light blue. The data set itself ("p8483.demo_1") and the variable name ("age") are in black. The comment I've put after the histogram request is color-coded green. Had I written something incorrectly or omitted a semicolon, SAS would very likely (though not always, depending on the type of error) have flagged the error by coloring the syntax red.

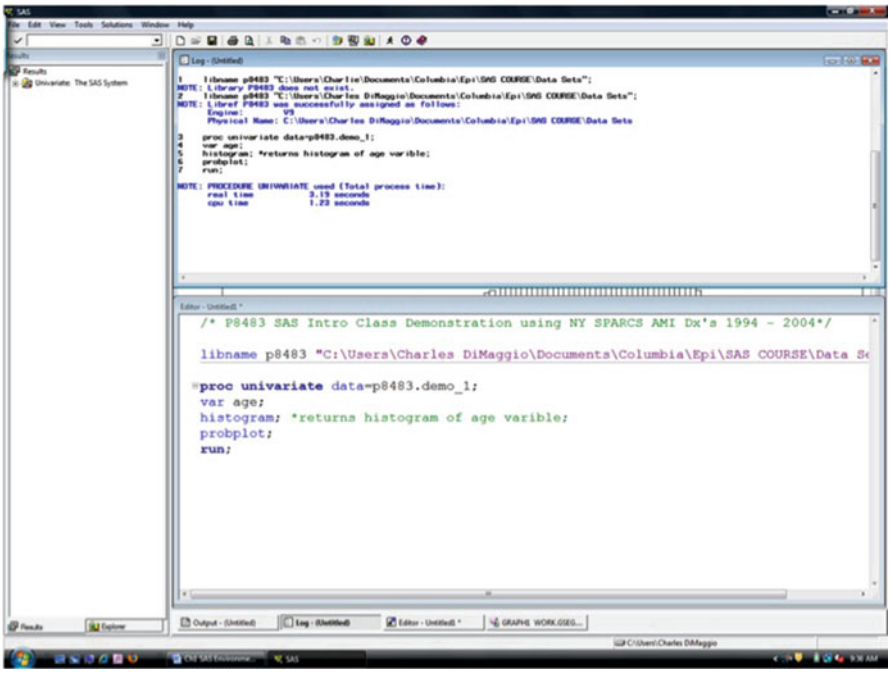


Fig. 2.2 The SAS Editor

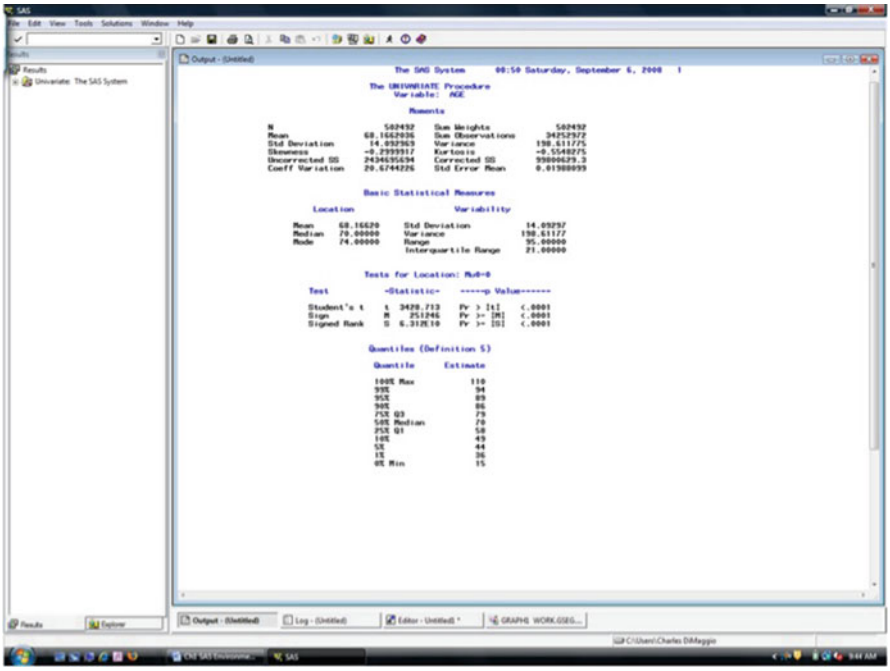


Fig. 2.3 The SAS Results Window

The log window on top gives some information about the syntax I've submitted and how long it took SAS to run the program. If there had been an error message, it would appear here in red. Also, note that the results pane on the left lists the univariate procedure I requested.

If I click on the Output tab at the bottom of the screen, I bring up my results (Fig. 2.3). In this case, I requested both statistical analyses and a graph (Fig. 2.4).

2.6 Two Types of SAS Programs

There are 2 basic types of SAS programs. You may hear them referred to as "steps" by old (and not so old) SAS Pros. They are the DATA step and the PROC step. DATA steps create or manipulate SAS data sets. PROC steps process data sets and conduct analyses. SAS knows a step is beginning when it sees DATA or PROC and knows one is ending when it sees RUN, QUIT, or another DATA or PROC statement. Read this paragraph again. Commit it to memory. It is the center around which all things SAS revolve.

It's helpful to know that DATA steps execute each line of your syntax for each observation in your data set before looping to the next line of syntax. SAS sees only

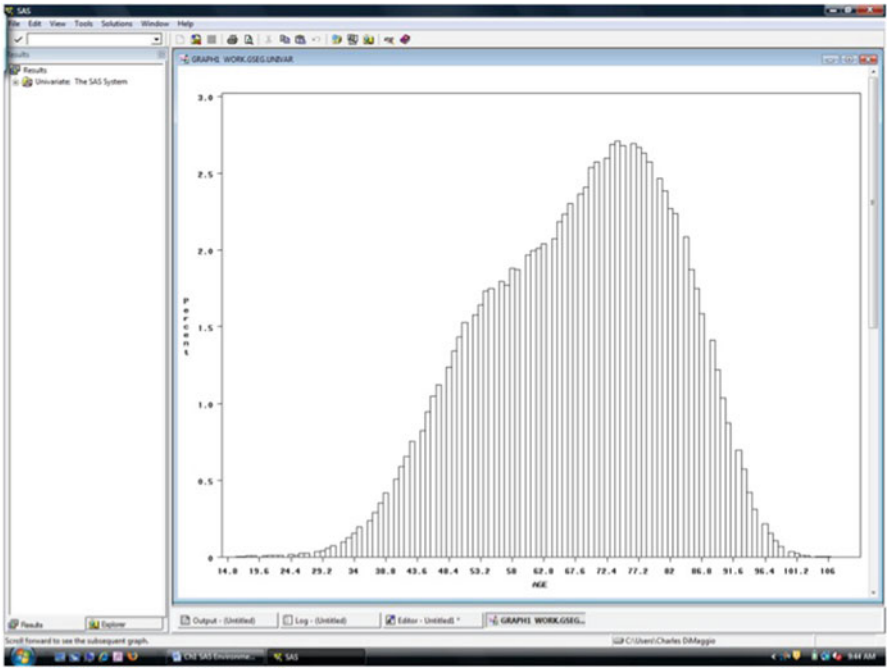


Fig. 2.4 The SAS Graphics Window

one observation of your data at a time and won't go onto the next until finished with first. This may not seem very useful information right now, but when we move on to consider various ways of reading data into SAS, it will help explain some inscrutable DATA step error messages.

As mentioned, SAS needs a RUN statement at the end of a program. But it doesn't really need one for each and every PROC. You can, if you like, string together a bunch of PROCS and then put a RUN statement at the end of them so they all run. I don't recommend this approach. I find the RUN statements a useful way of breaking up long or complex SAS code, making it both easier to read and debug.

SEMICOLONS

In English, a semicolon is used to link two independent clauses and is considered by some to be obsolete; I like them. In SAS, the semicolon is used to demarcate the end of a syntax statement; SAS likes them very, very, very much.

2.7 Two Kinds of SAS Data

Now we know there are (basically) two kinds of SAS programs. There are only two kinds of SAS variables: character and numeric. This distinguishes SAS from other programs that require many specific variable definitions. Even SAS dates (which we'll spend more time considering in later chapters) are numbers. They are based on a reference to January 1, 1960. (Why? Your guess is as good as mine.) So January 2, 1960, is the number 1, December 31, 1959 is the number −1, January 3, 1961 is the number 368, and so on. Missing character variables are denoted by a blank space; missing numbers are denoted by a decimal point.

2.8 Two Parts to a SAS Data Set

A SAS data set file holds (1) the data itself and (2) the descriptor portion that describes the variables.

The data portion of a SAS data file consists of the rows and columns of the data itself. You can view the data itself a couple of ways. You can write a *PROC PRINT* command in the Editor window. Or, you can double click on the physical file in the library folder found in the Explorer window.

The descriptor portion is like meta-data. It contains information on every variable such as the type of variable it is (we now know it can only be numeric or character), its length, any format you've applied, and any label you've applied. We'll discuss formatting and labeling later on. You can access this descriptor information using a *PROC CONTENTS* command.

SAY GOODBYE TO YOUR SPREADSHEET

Perhaps one of the most difficult aspects of the transition to SAS from programs such as Excel and SPSS is letting go of the need to manipulate a physical spreadsheet. SAS data seems to be floating in some netherworld to which you are sending commands.

In fact, SAS data sets very much do exist in a physical form on your hard drive. You just do not have call to work on the physical file directly. This is an asset when you are working with the large files for which SAS is so well suited. It is easy enough to find and call up the file when you have a handle of the SAS filing system of libraries. You can, at any time call the data file up directly. If only to reassure yourself that it exists.

2.9 Some Simple SAS Utilities

Printing and saving in SAS works pretty much the way it does in any Windows program. The default output file format is something called a "list" file, but you can also save your output as file types compatible with Windows Office programs such as Rich Text Format (RTF) for Word text documents and Enhanced Metafile (EMF) files for inserting graphs and charts into windows documents. SAS also has a nice export function, so you can send your data sets to programs like Excel. To print or save just part of your output, highlight the results in the output window, right click, and print, copy, paste, or save.

You can change your system options for things like lines per page, font type and size, centering text under the tools menu:

```
Tools --> Options --> System
```

2.10 Getting Help

OK, SAS is not exactly world renowned for the ease and clarity of their help and documentation. It sometimes seems like you need to know the answer to find the answer. The situation, though, has improved considerably over the years. You can access information on most procedures under the Help menu. You need, though, to go through a couple of steps:

```
Help --> SAS Help and Documentation --> SAS Products
--> Base SAS --> SAS Procedures --> Procedures
```

You can get list of SAS functions (we'll discuss them later) at:

```
Help --> SAS Help and Documentation ---> SAS products
--> Base SAS --> SAS Language Dictionary -->
Dictionary of Language Elements --> SAS Data Set Options
```

(whew)

You could, alternatively, type "help" and the name of the procedure you want help for (e.g., "help univariate"), in the Command Window. The Command Window is located on the upper left-hand part of your screen and has a little check mark icon next to it. You can also get a list of SAS keyboard shortcuts ("function keys") by typing "keys" in the command window. This would tell you, for example, that F8 is the keyboard approach to submit a command.

Remember, there's (almost) always more than one way of doing things in SAS. It is an extraordinarily robust and multifaceted computing language. For example, to clear a your output screen, you can type "clear" in the Command Window or use the short-cut function "CNTRL-E" One of the nice things about SAS's popularity is

that there is invariable someone who faced the same issue you have, and often, they were nice enough to post something online about how they solved their problem. I answer most of my questions with Google.

Problems

2.1. Submitting Commands and Reading Output

Start up SAS and with the Editor window active, use the "File" menu to navigate to your files. Open the intro–sparcs–ami syntax file by double clicking on it. Scroll down to the code titled "compare ages".

Type in the full file-path location of the data file

```
demo_1
```

in between quotation marks following the statement

```
libname p8483
```

Don't forget the semicolon. Submit the t-test statement comparing the age of non-New York City acute myocardial infarction patients ("upstate") to residents from New York City. What is the mean age of upstate AMI patients? NYC AMI patients? Is this difference statistically significant? Do you think it is clinically important?

2.2. Saving SAS Output to Microsoft Word

Clear your output screen by pressing Cntrl-E. Do the same with your Log screen. Using the intro–sparcs–ami syntax file, run the univariate analysis of age. Save the output of the univariate analysis as a rich text (RTF) file on your desktop, by

1. Saving the RTF SAS file as a MS Word file
2. Using "File - Export" to export your histogram onto your desktop as a Windows Enhanced Metafile (EMF)
3. Opening your saved MS Word file created in step 1 and inserting your histogram in it using "Insert–Insert Picture"

2.3. Using SAS Help

Use the SAS help system to find an example of how to use the "OBS=" data set option to limit the number of observations SAS processes. Copy and paste the example from the help file into the word document you created in Exercise 1.2.

2.4. Using PROC CONTENTS to View Variables

Return to the intro-sparcs-ami syntax file in your editor window. Write in and submit the following syntax:

```
proc contents data = p8483.demo_1;
run;
```

How many observations and variables are in this data set? Is the data sorted? What type of variable is the DATE variable? What type of variable is the DISPO variable and how long is it?

Use the SAS Explorer (hint: double click on the libraries folder) to find the demo-1 file on your computer system. Double click it to open it. Scroll down and inspect the DATE and the ECODE variables. How does the DATE variable appear? Does the ECODE variable look to be very useful?

Close out of SAS. You do not need to save your changes.

Chapter 3
Working with SAS Data

Abstract This chapter introduces the core concepts of SAS programming. We learn how to use SAS "libraries" to locate, refer to and save data files. How to crate data files by inputting data directly into the Editor. How to read in existing data files by specifying data sources and the kinds of data they contain. And, how to decipher error messages and debug some common problems. The chapter ends with some notes on a few tricks for getting data into SAS, and the SAS utilities available for reading in MS Excel files.

3.1 SAS Data Libraries

SAS data files are stored in *libraries* of similar or related data sets. New SAS users sometimes find libraries confusing. You can think of them as a sort of shorthand or pointer SAS uses to find files on your computer. SAS doesn't create the folders themselves. You do that in Windows (or Mac's Finder) as you normally would. SAS just has a unique way of referring to those folders, and it provides you with a way to direct your work to the files that are stored in that folder. The names SAS uses for the folders on your computer are called *librefs*, and you assign the names for your folders using a.

```
LIBNAME
```

statement.

For example, say you will be creating and working with a number of SAS data files from New York State hospitalization discharge (SPARCS) data. In Windows, you can create a folder on your desktop named "sparcs" to keep them in by (1) right clicking on your desktop and (2) following the prompts for a new folder. If you then right click on this new folder and check its properties, you will be told it is located on your hard drive in a location called (for example)

```
'C:\windows\desktop\sparcs\'
```

C. DiMaggio, *SAS for Epidemiologists: Applications and Methods*,
DOI 10.1007/978-1-4614-4854-9_3,

Notice this is not a file. It is an empty folder you created in Windows.

Now, in SAS, before you start creating data sets based on the SPARCS data, you will use a

```
LIBNAME
```

statement to create a libref called "sparcs" that points SAS to the sparcs folder you created in windows. From now on, in SAS, you will not refer to the file with the Windows location of

```
'C:\windows\desktop\sparcs\'
```

You will instead use the libref name ("sparcs") you created with your LIBNAME statement. Physically, the file is in

```
'C:\windows\desktop\sparcs\'.
```

The SAS libref is just a shorthand name for that location.

The libref (which refers to a folder) is paired with a data filename, separated by a dot, to uniquely identify a data set ("libref.dataset"). This is a key concept and bears repeating: In SAS, data files are referred to by their "libref.dataset" name. So, rather than writing syntax to tell SAS to conduct analyses on a mortality file called

```
'C:\windows\desktop\sparcs\
mortality.sas7bdat',
```

[1] you refer to your mortality data file in SAS as "sparcs.mortality." This filing system is another feature that allows SAS to work across many types of computing platforms. After a little getting used to, you will find this kind of shorthand actually quite useful and logical.

Finally, it's worth restating (because it seems to be a stumbling block for folks) that the folder to which you direct SAS in the libname statement must physically already exist; SAS won't create it.

STOP. NAME YOUR LIBRARY.

You need to reissue the LIBNAME reference *each time* you start a SAS session. If you want to avoid this, you can R click libraries in Explorer, choose "New," click "Enable at startup," name it, and locate it by browsing to the directory.

3.2 Two Special SAS Libraries

There are two special libraries you should be aware of.

[1] Sas7bdat is the file extension for SAS data files.

WORK is a default library that only exists during that particular session. It need not be invoked at all. If you reference a data set without specifying a libref, SAS will automatically store it in the work library.

Caution: The work library is erased at the end of the SAS session! This is useful if you are, say, working on creating a data set, and there are a number of intermediary steps that you don't need to hold onto. But remember, if you want to keep a permanent data set, it must be stored in a permanent library, *not* the Work library. And remember, a single named SAS file implies (invokes) the work libref which is temporary.

SASUSER is a folder location that comes preinstalled on Windows versions of SAS. It is a folder in the bowels of SAS that you can invoke without first creating it. It is an easy way to save a data file to a permanent folder if the whole libref/LIBNAME thing is just too overwhelming.

3.3 Three Ways to Browse SAS Data Libraries

To explore your SAS data libraries, you can (1) type libname in the command window then drill down to files with your mouse, (2) click on the Explorer Window tab then double click on Libraries, and (3) write a PROC CONTENTS command in the Editor Window with an ALL keyword to invoke all the files in the library and an NODS option to suppress the descriptor portions of the files. The syntax looks like this:

```
proc contents data=sparcs._all_ nods;
     /*note the space between _all_ and nods*/
run;
```

SAS will print out all the SAS data sets within the library and give some information on each file (Fig. 3.1).

3.4 Inputting Data into SAS

This is where SAS excels (pun intended). You can, essentially, read almost any data type into SAS, and there are myriad ways of doing it, depending on the format and location of the data. For any approach, though, you need to first indicate to SAS that rather than performing a procedure on an existing data set (with a PROC statement) you are creating or manipulating a data set (with a DATA statement). You then specify the name of the data set, whether it should be stored in a temporary (WORK) library or in a permanent library, and how the data should be INPUT into the file by specifying (in the order in which they appear in the data file) the variable names and what type of data they are (numeric or character).

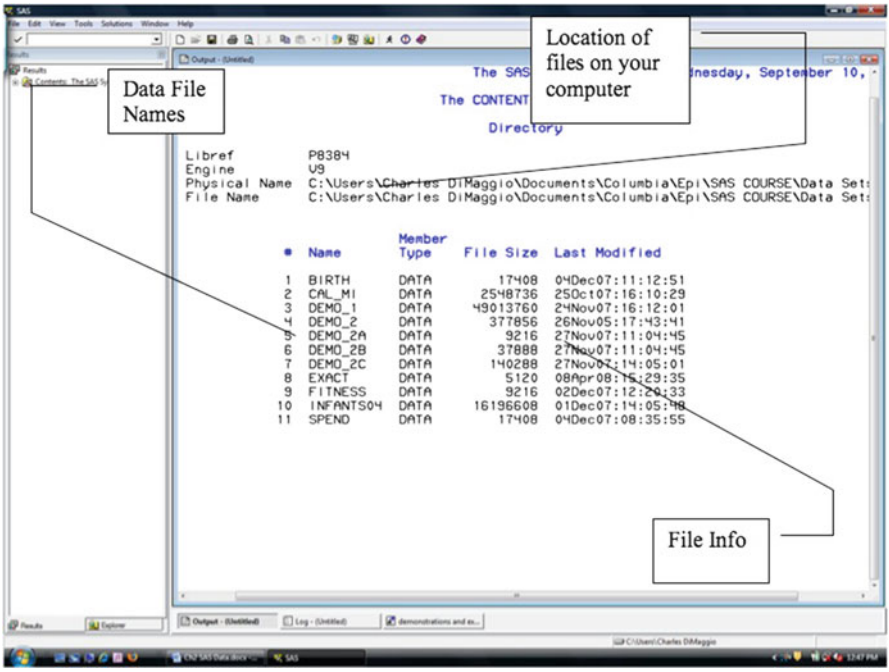


Fig. 3.1 PROC CONTENTS Output

The first essential step in reading in data, though, is to determine how your data is laid out.

In general, and certainly for epidemiological purposes, it makes sense for each line or row of data to correspond to an individual or an observation on an individual, with variables for that individual observation arrayed along that row or line of data. If you are typing in or creating your own data set, this may be the most straightforward approach.

ID	Variable 1	Variable 2	Variable 3
Person 1	AAAAA	BBBBB	CCCCC
Person 2	DDDDD	EEEEE	FFFFF
Person 3	GGGGG	HHHHH	JJJJJ

But, you may be working with data from another source, perhaps from a government administrative health data set, and what works for epidemiologists and SAS programmers is not always the best approach for data base managers. Especially with large data sets, space and when file size considerations are important.

The data may be arrayed as a single continuous array of values:

```
AAAAABBBBBCCCCCDDDDDEEEEEFFFFFGGGGGHHHHHJJJJJ
```

There may be an unbroken row of data for each observation with a hard return or a character at the end of each observation:

```
AAAAABBBBBCCCCC
DDDDDEEEEEFFFFF
GGGGGHHHHHJJJJJ
```

The data may be arrayed in columns with a space between each column. That space may not even really be a space, but rather a tab character:

```
AAAAA  BBBBB  CCCCC
DDDDD  EEEEE  FFFFF
GGGGG  HHHHH  JJJJJ
```

The variables may be separated by some other character such as a decimal point; they may continue on another line or even on another file. Missing observations may be represented by a series of blank spaces or a special character or series of numbers or two tab characters or may not be indicated at all.

The data may also come in any number of formats. You may be one of those fortunate individuals who receive his or her data in SAS format, but this is not usually the case. It may be in ascii or text format (indicated by a .txt file extension). It may be in Excel or SPSS or STATA or R format.

The good news is that SAS can handle just about any data set in just about any configuration. It may take some work, though. It's been my experience that the majority of data analytic effort in the secondary epidemiologic analysis of data is spent in wrestling the file into an appropriate format. While most external data sets come with information describing how they are laid out (we'll talk more about that shortly), the best first step is simply to look at the raw data.

LOOK AT YOUR DATA.

The potentially complex and myriad variety of data set ups, leads to the first and foremost rule of data entry: *LOOK AT YOUR DATA*.

Usually, the best first step is to use a simple text editor, like Notepad (or for larger files Wordpad), to open up the file.

There are two (main) ways of reading data into SAS: You will generally either type (or cut and paste) data directly into SAS or read in an external data file.

3.5 Reading in Data from the Editor Window

In this, most simple, approach, you either type or paste data directly into the SAS editor window. For many smaller epidemiological analyses involving less than, say, a few hundred observations, this may be all you will ever need. The steps are as follows:

1. In the data editor window type the word DATA. Then name the data set being created or manipulated in the libref.sasdataset format and end the line with a semicolon.[2] If you are creating or manipulating a temporary data set, you can use the work libref omitting the libref altogether. If SAS doesn't see a libref in the DATA step, it will assume it is a temporary work data set.
2. On the next line of syntax, type the word INPUT. You will then describe to SAS how to read in or input the data from the raw data by naming the variables, specifying locations, and identifying them as character or numeric. Your input statement will be dictated by how your data are laid out (see below). This line of syntax is also ended with a semicolon.
3. On the next line of syntax, type the word CARDS or DATALINES followed by a semicolon. This indicates to SAS that what follows next is the raw data itself.
4. Type in or paste the raw data. Place a semicolon at the end of the data, on a line by itself. This indicates you have finished naming the data. You will know the data is successfully entered when it is highlighted a pleasant yellow color.

NAMES

SAS is a little fussy about how you may name data sets and variables. Both data set and variable names must start with character or underscore. They can be 32 characters long with no spaces or special characters like # or %.

CARDS

Cards may seem like an odd choice of word to indicate data. It refers to those days of yore when data observations were entered on punch cards.

3.6 Two Basic INPUT Statements

As noted above, the INPUT statement is where you tell SAS how your data is laid out. The majority of data can be input in one of two basic ways: as fixed columns or with user-specified formats. The second approach, where you specify the location and type of each variable, is the most flexible and probably the safest approach. But, for completeness sake, and in the case you find a nice neat data set, we'll consider the first approach.

[2]Remember, you must already have created the libref with a LIBNAME statement, or you will get an error message.

3.6.1 Space-Delimited and Column Input

For simple space-delimited data, where each row represents an observation, all the variables for each observation line up nicely, and there are no missing values, it will often suffice to just INPUT the list of variable names. Note that the default SAS variable type is numeric, so if you have character variables, you have to alert SAS by putting a $ sign after the variable name to indicate it is a character variable.

The following INPUT statement indicates that each row represents an observation containing three variables. Variable 2 (named var2) is a character variable.

```
INPUT var1 var2 $ var3 ;
```

You will run into problems with this approach if there are any missing data. SAS will just read the data consecutively and place them under the wrong variable.

If you know or suspect there are missing variables, you can add an additional level of control by explicitly telling SAS where each variable column begins and ends by specifying the number of spaces after the variable name. In the following syntax we take the previous input statement and tell SAS that variable 1 (named var1) begins on the first space and ends on the 5th space; there are then two blank spaces that SAS should ignore, and variable 2 begins at space 8 and goes till space 10 and that there are another two empty spaces before variable 3 begins at space 13:

```
INPUT var1 1-5 var2 $ 8-10 var3 13-26 ;
```

Here's a demonstration of how you might use column input with data entered directly into the editor window to create a file of discharge data on a group of 4 patients (Fig. 3.2). The variables are the patients' first and last names, their social security number, and whether they were admitted or released. Perhaps you've abstracted them directly from patient charts. Look closely at the syntax in the bottom editor window. You should understand what each line of code means. If you don't, reread the section above before proceeding.

Once you've created a data set, look at it. There are two ways to do that. You can either use a PROC PRINT command, as I did in the syntax above. (Notice I didn't have to specify the data file to print because it is the only data set created during this session). You will get an output screen that looks like this (Fig. 3.3).

Or you can navigate to the work library and double click on the data file name to open the viewer (Fig. 3.4).

One (very) important final point about simple column input: it is a poor choice for anything other than standard numeric or character variables. So if your data contains dates, or even just commas separating numbers, you are better off with user-defined input.

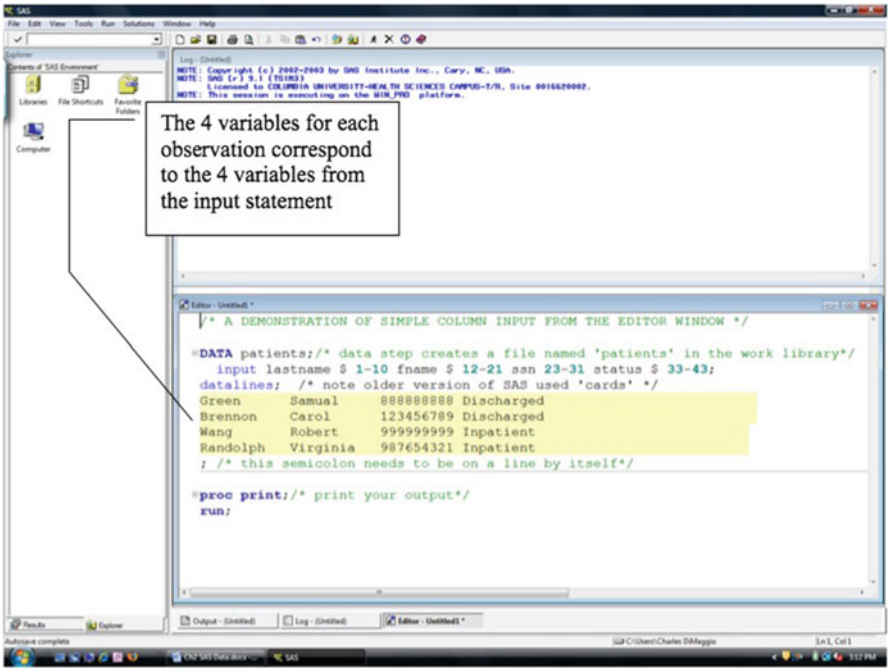


Fig. 3.2 Entering data in column format directly in the editor window

3.7 Reading in Data from External Files

If you plan on working collaboratively with other folks' data, or if you will be analyzing standardized health data sets downloaded from government sites such as the CDCs National Center for Health Statistics, entering data directly into the editor window with a "cards" or "datalines" statement is not an option.

More often than not, you will be working with external data sets formatted in ways that make sense to the originator and, very likely, to no one else. In this case, rather than specifying "cards" or "datalines," you will need an INFILE statement to identify the file location and type. And, most likely, you will need to create some user-defined inputs, rather than use simple column input.

3.7.1 The INFILE Statement

An INFILE statement identifies the location and type of the raw data file you want to read into SAS. After typing the word INFILE, type, within quotation marks, the

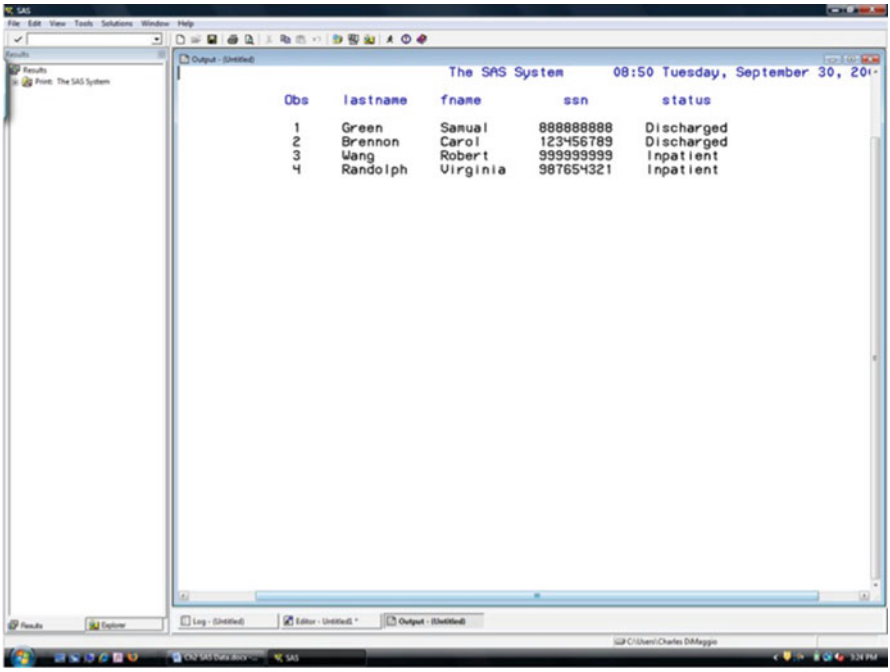


Fig. 3.3 Printing out your data

path, on your computer, to the file. Make sure you specify the file extension and end the statement with a semicolon:

```
INFILE c:\temp\sparcs.txt;
```

You can put various options into the INFILE statement to help SAS read the file. Three common ones are as follows:

1. *LRECL=n* stands for logical record length and tells SAS how long each row or observations is. The default SAS setting is that each row or observation is 256 characters long, and then a new row or observation begins. If each row of data is shorter than that, you do not need to specify the record length. But, if it is longer than 256 characters, you need to use the LRCL option, or SAS will truncate your data at 256 characters. Remember, SAS includes blank spaces when it calculates the number of characters in a row.
2. *OBS=n* tells SAS to read in n number of observations. There is no preset number of observations, and by default, SAS will read in all your observations. This option is useful if you are troubleshooting a large data set that takes a long time to read and want to debug your DATA syntax without having to read the entire data set.

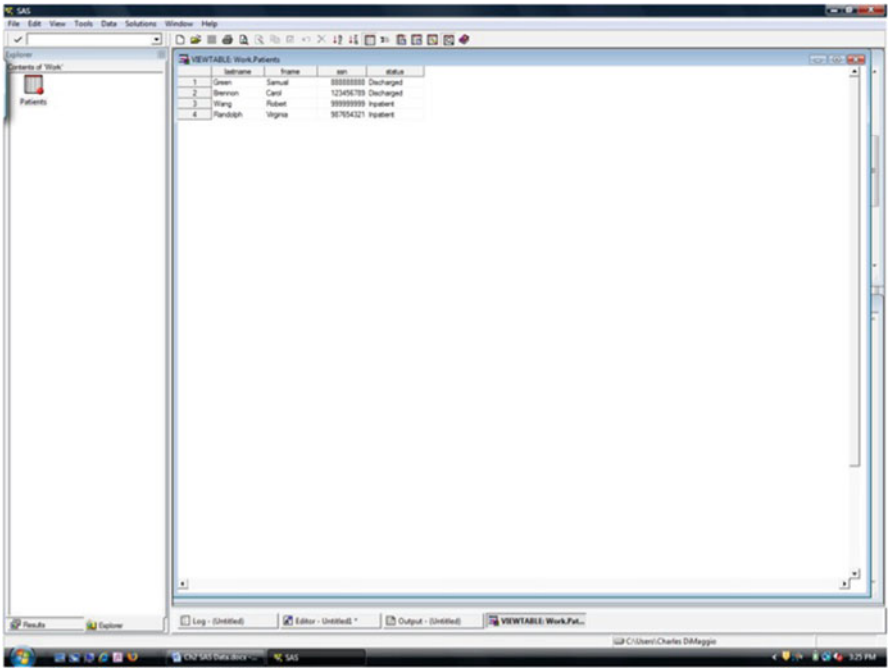


Fig. 3.4 Viewing data using the SAS Explorer Window

3. *MISSOVER* assigns SAS missing data values to variables not given a value in a data record. (Well talk more about that later.)

The following line of INFILE syntax tells SAS that the data is a text file (the .txt extension) called ADR02NY located on your computer's "F" drive in a folder called "SPARCS" which is a subfolder of "K Data", that each observation is 450 characters long and that it should only read in the first 100 observations:

```
INFILE 'F:\K Data\SPARCS\ADR02NY.TXT' LRECl=452 obs=100;
```

3.7.2 Formatted INPUT of External Data Sets

Recall that with simple column inputs, where you just list the variable names, whether they are character variables, and perhaps where the columns begin and end, is useful for the simplest data sets. If your data is not in some form of space-delimited column format, or if it contains any data other than simple numbers and characters, you will need to further define the variables in the INPUT statement.

After typing INPUT, for each variable, you specify a location by using a *pointer control*, and then you specify the informat for that variable.

The pointer control is written as

```
@n
```

and tells SAS the variable begins at column n.

3.7.3 Informats

Informats, when used as part of an INPUT statement, tell SAS how to read in and store data by specifying the width of the variable field and what kind of variable it is.

3.7.3.1 Anatomy of an Informat

SAS INFORMATS have the following form:

```
<$>informat-namew.<d>
```

where

- *$* indicates a character variable. If you do not explicitly specify the type of variable or omit the $, SAS assumes it is a numeric variable.
- *Informat-name* names the informat. SAS has many predefined informats for things like characters that retain leading blanks, characters that trim leading blanks, dates, dates and times, comma-separated numeric values, percents, etc. However weird the format of your data, it's likely SAS has an informat for you. You can look up informats in SAS documentation, but I have found that the two best sources of informats are Delwhiche and Slaughter's "The Little SAS Book" for more common informats and the Internet for more exotic informats.
- *w* is a number that defines the total width of the field to read.
- . is a required delimiter and indicates to SAS that this is an informat. You will note when using the enhanced program editor (which you should be using anyway) that when you enter the dot as part of an INPUT statement, the informat turns a mellow green color.
- *d* placed after the required dot delimiter defines the number of decimal points for a numeric variable and is optional.

Some examples of SAS predefined informats are as follows:

- *$CHAR6.* defines an 8-space-long character variable that retains leading and trailing blank spaces.
- *$6.* defines an 8-space-long character variable that trims leading and trailing blank spaces.
- *4.* defines a 4-space-long numeric variable with no decimals.
- *4.2* defines a 4-space-long numeric variable with 2 decimal places.

- *COMMA4.2* removes embedded commas and dollar signs from a 4-space-long numeric variable with 2 decimal places (In general, *COMMAn.* reads n columns of data and removes nonnumeric characters like commas and dollar signs. It affords much more control over the data input and can deal with most any data type.)
- *MMDDYY8.* defines an 8-space-long date variable in a month-day-year (mm/dd/yyyy) format.

As a demonstration of reading in an external data file, let's return again to the idea of hospital discharge data. We'll be spending a lot of time working with such a data set to illustrate SAS concepts and procedures , and we'll define it in more detail in later chapters. For now, say you have a text file of all hospital discharges in a certain area. You are interested in creating a SAS data file of the first 1,500 observations that contains the patients' ages, zip codes of residences, gender, race, ethnicity, and primary diagnosis.

Review the following syntax. You should be able to recognize all the elements of the syntax. Note that, as in the previous simple column input from the text editor, we begin with a DATA statement that creates a temporary file.[3]

Also note the reassuring blue-colored notes in the log that tell us that file was successfully created (Fig. 3.5).

3.8 Behind the Scenes of a Data Step

It's been my experience that most problems using SAS occur during the creation or manipulation of data sets. SAS will do what it can to tell you what went wrong, but a basic understanding of what is going on behind the scenes is helpful in deciphering error messages.

To begin, DATA starts the process and names the file. The *INFILE* statement then creates what is called an *input buffer* to temporarily hold each observation as it is being processed. The *INPUT* statement then takes values out of the input buffer and stores them in what is called a *program data vector* (PDV) which associates attributes such as length, type, and name with each variable in an observation. The *RUN* statement then creates the descriptor portion of the data set from the PDV.

I realize that's quite a mouthful, and while it is not strictly necessary to know all these steps, it is very helpful to be aware of how the overall process occurs. In particular notice how the process loops from observation to observation. After the RUN statement, SAS automatically returns to the beginning of the DATA step, reinitializes all the variables in the input buffer to missing, then INPUT reads the variables from the buffer to the PDV again, and then RUN creates the descriptors.

[3]What is the name of the file we are creating? How do you know it is temporary? In what library would you find it?

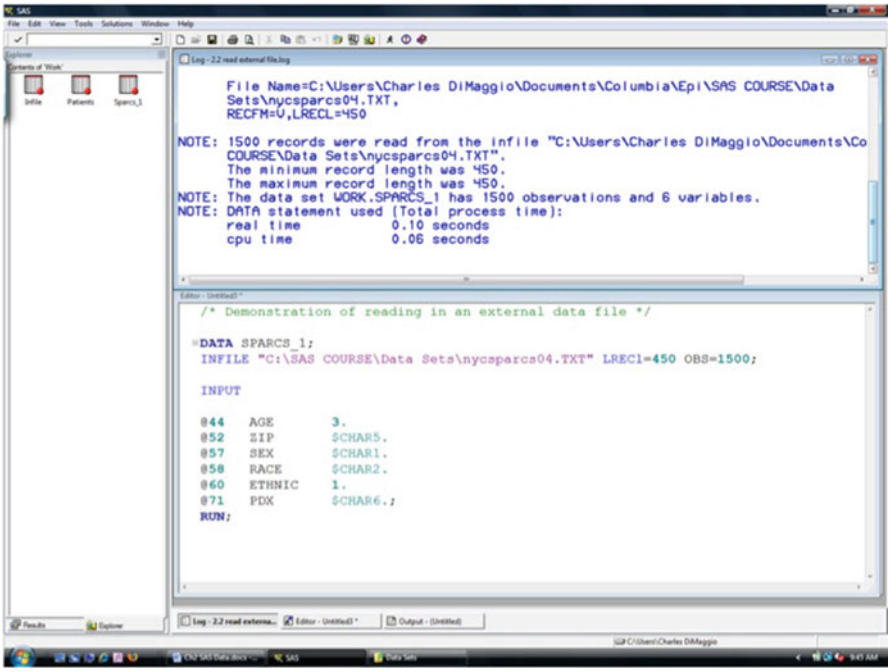


Fig. 3.5 Reading in data from an external file

Finally, the process loops back again to the DATA step. It helps to know that your data file is processed observation by observation.

Being aware of this process may help you better understand the invariable (and sometimes inscrutable) error messages SAS will send your way.

3.8.1 Deciphering Error Statements

If SAS encounters an error in either the syntax or in the data set, it will let you know through an error message in the log. *SAS may or may not stop processing the data set.* This is an important and potentially crucial point. It is of more concern when a data set when contains errors is created. Read your log. Always.

3.8.1.1 The Most Common Errors

The most common error is leaving out a semicolon somewhere along the way. Because the omission of a semicolon can result in most any kind of error, the error message in the log usually won't state that you omitted a semicolon. Generally,

unless the error message clearly indicates something else, always look for a missing semicolon first.

After that, the most common errors in reading data are simple syntax errors such as misspelling a data file, leaving off an extension, or leaving off a quotation mark. Note that if, for example, you leave off or misspell a data file extension, the SAS error message will not be "you misspelled the extension," SAS will, rather, tell you the source file doesn't exist.

After syntax misspecifications, SAS will most often report errors when it comes across missing values or finding a character value when it is expecting a numeric value because the INPUT statement defined a numeric value for that variable.

3.8.2 Error Messages

You will find error messages in the log.

An error message may look something like this:

```
_ERROR_=1 _N_=10
```

3.8.2.1 Anatomy of an Error Message

- *Number 1* in the error statement is a binary flag SAS uses with each loop of the data, with 1 meaning a data error was encountered and zero meaning no errors.
- *The number 10* indicates that the error occurred during the 10th loop through the data, which, if the data file is specified correctly, should be your 10th observation.

You will also get an indication of exactly at what column SAS encountered the error, what the error was, and (often) what SAS was expecting.

In the following screen shots, we will look at some error messages you might get when trying to read in data (Fig. 3.6).

Here, SAS gives us the error message that the "file does not exist." We are then informed that SAS "stopped processing this step" because of this error, that the data set we were trying to create "may be incomplete," and that (helpfully) if there had been an existing such data set, it was not replaced. All very useful information, but what did we do wrong?

The most important bit of information is that SAS believes the file to which we referred does not exist. This should lead us to look at the part of the syntax that refers to the file. This would be our INFILE statement:

```
infile "C:\Users\Charles DiMaggio\Documents\K\P6781\Thumb
   Drive\ F_CALIFORNIASAS\Hertt95.txt" lrecl=999;
```

First off, SAS is (almost) always right. It is looking for the file you specified, and it is not there. Basically one of two things has occurred. You correctly typed the file

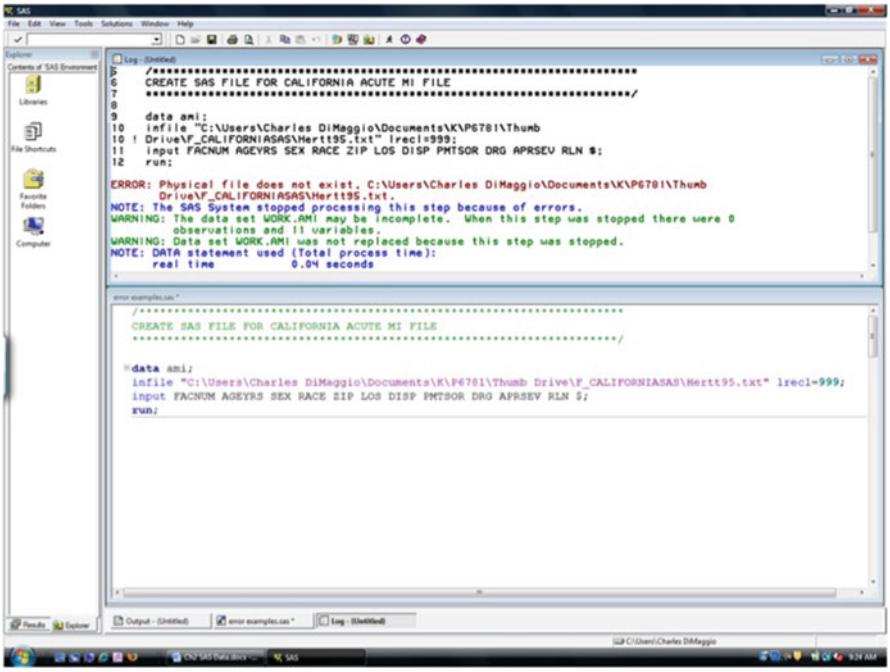


Fig. 3.6 Error messages when reading in a data message

name, but it is either not on your computer or in the folder you specified. Or, you typed the name incorrectly. This is the more common situation. Here, I typed

```
\Hertt95.txt
```

when the actual file name is

```
\Heart.95.txt
```

You can avoid this kind of error. When you write your INFILE statement, locate the file to which you want to direct SAS by using Windows Explorer, then right click the file name, select properties, and copy and paste the file location and name into SAS. This should result in fewer typing errors.

The next screen shot demonstrates a second common error (Fig. 3.7). Here, I present only the log screen. You will see that I've corrected the file name, and SAS now reports (in reassuring blue font) that the infile statement is the same as the file name it found.

The error now is of the somewhat inscrutable

```
_ERROR_=    _N_=
```

type. SAS tells us that it encountered invalid data for a variable called RLN. It is occurring on every pass of the data and seems to be at position 30.

The most appropriate next step is to look at our data in a simple text editor and compare what we find at position 30 with what we told SAS it should find.

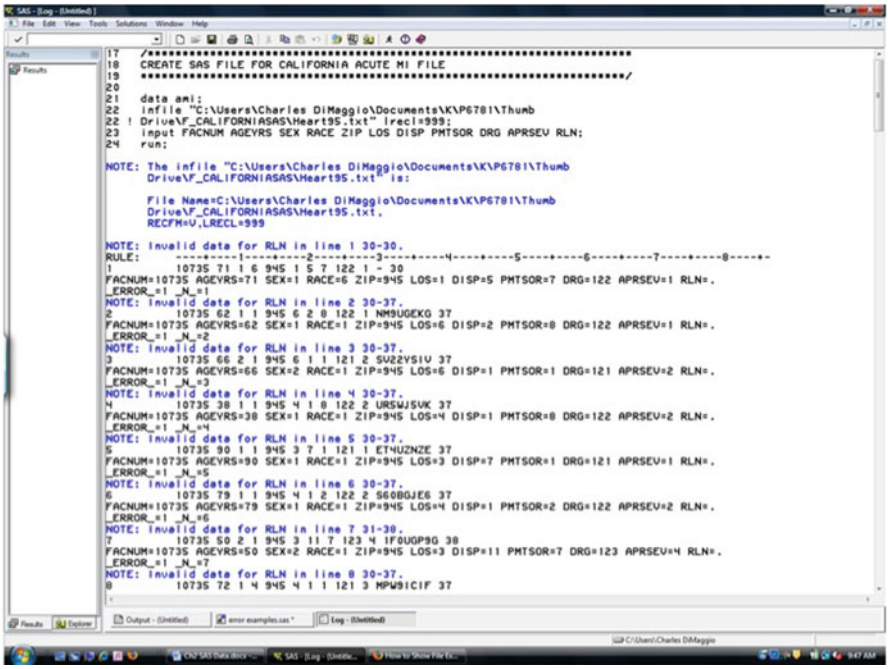


Fig. 3.7 Log screen for errors reading in data

Here is a screen shot of the first few lines of the text file we specified in our INFILE statement (Fig. 3.8). If we count 30 spaces from the first column, on the first line, we encounter the character

–

On the second line, we encounter the string of characters

```
NM9UGEKG
```

We can now look at our input statement to see what we told SAS to look for:

```
input FACNUM AGEYRS SEX RACE ZIP LOS DISP PMTSOR DRG APRSEV
RLN;
```

We see that we specified the relevant variable (RLN) as numeric. When SAS, instead, encountered a character variable at that position, it returned an error message.

The next screenshot indicates an even more problematic situation (Fig. 3.9). This is a continuation of the same error message. First we see that after 20 error messages, SAS reaches its limit and informs us that, although there may be more of the same errors, it will no longer document each of them. Of greater concern, though, is that SAS created the data set, errors and all. Note that SAS completed a PROC CONTENT without any error message.

This example illustrates the crucial importance of reviewing your log, particularly when creating data sets. While you should always look at your log, it is

Fig. 3.8 Reviewing the raw data

critically important to look at the log every time you run a data step. If you look at it after running some procedures, you may not catch the error.

3.8.3 A Few Other Common Errors

As noted above, the most common error is probably omitting a semicolon. It is very unlikely, though, that you will get an error message from SAS telling you that you omitted a semicolon because SAS will read what follows the omitted semicolon as part of the statement preceding the omitted semicolon. This can result in most any error message.

Another common error is forgetting to put a RUN statement at the end of your syntax or (again) forgetting the semicolon after the word run. This basically puts SAS into a state of suspended animation. It's like it knows what you want it to do but is just waiting for permission to go ahead and do it. You will see the word "Running" in the title bar of your output and log windows. You don't need to resubmit your entire program. Just type "RUN;" and submit that. If all you did was omit the semicolon after the word RUN, just highlight and submit the semicolon.

Perhaps not as common as omitting a semicolon or a run statement, but markedly more annoying, is submitting unbalanced quotes, i.e., omitting a quotation mark at

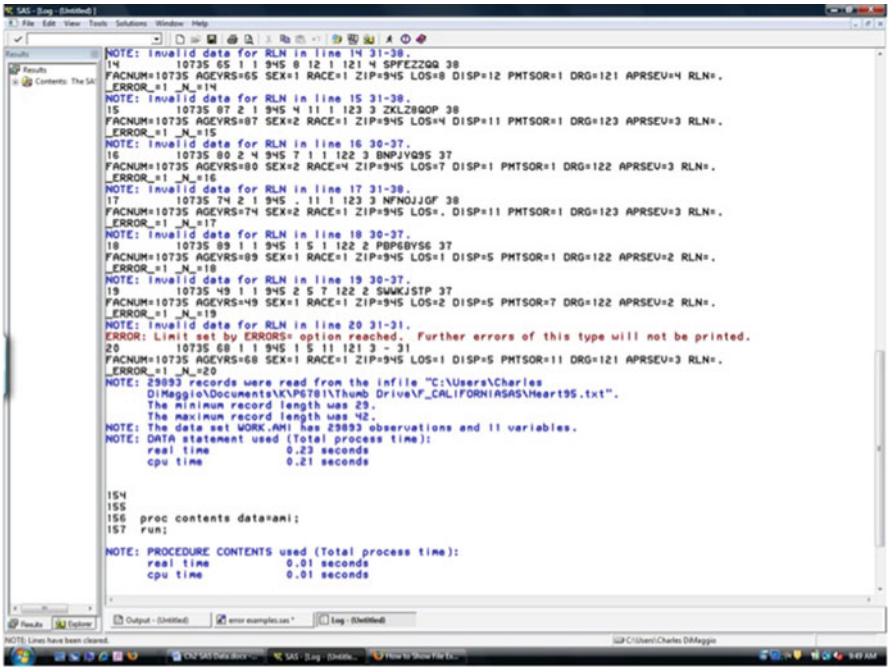


Fig. 3.9 Creating a data set despite errors

the beginning or end of a string of character values. SAS will see everything after the lone quotation mark as part of a character string. If you just recall and resubmit the code, SAS will just see it as a further appending to the original code. You need to give SAS what it's looking for: a quote mark to end the first quote. A neat approach, taught as part of the SAS Institute's introductory course, is to type

```
*run;
```

In case it is not a mismatched quote, the * at the start creates a comment that SAS doesn't read.

3.9 Notes on Manipulating Data (or How to Tame an Annoying Data Set)

SAS offers many, many ways of inputting data. I have never come across a data set it couldn't handle. Here are a couple of features and tricks that I've found useful. You can find a lot more with some simple online searches and in books like the admirable little Delwiche and Slaughter.

3.9.1 Illogically Arrayed Data

Your data may not always be logically arrayed with one row of data corresponding to one observation.

3.9.1.1 More Than One Row per Observation

You can read more than one row of data into a single observation by using a slash (/) to tell SAS to skip to and continue reading the next line of data. For even greater control, you can use a #n to tell SAS to skip to a specific line of data (where n is the line number). Note that the #n is a more flexible approach. You can even tell SAS to go back to a previous line.

Say, for example, you had the following raw data indicating the name of a town, its population, the number of cases of some particular disease in that population, and the number of deaths among those cases:

```
Buford
1000 20
5
Tookton
2000 50
3
```

You could use the following syntax:

```
INPUT town $ / population cases / deaths;
```

If, alternatively, you just wanted a count of deaths for each town, you could input the data like this:

```
INPUT town #3 deaths;
```

3.9.1.2 More Than One Observation per Row

Alternatively, your data may be arrayed such that each row of data represents more than one observation. You can use @@ at the end of the INPUT statement to tell SAS to start a new observation regardless of whether it is at the end of a line of raw data or not.

As an example, say our town data above is arrayed like this:

```
Buford  1000 20 5 Tookton 2000 50 3

INPUT town $ population cases deaths @@;
```

If your data file has descriptive text at the beginning that you don't want to read in, you can tell SAS to start reading the file at some row other than the first row with FIRSTOBS = n as part of your INFILE statement (where n is the line of data at which you want SAS to begin).

Say, for example, you are reading a census file into SAS, and the first few lines are administrative text you don't need. The following INFILE statement will start reading in the data at the 26th row:

```
INFILE C:\Documents and Setting\Charles DiMaggio
       \My Documents\Census\ny_data.txt FIRSTOBS=26;
```

A very useful, INFILE option is to limit the number of observations you want SAS to read with OBS=n. As noted above, this is helpful if you have a very large data file and you want to debug or make sure your INPUT statement is free of errors.

This syntax will read in only the first 26 lines of data: (Note the difference from the previous example.)

```
INFILE C:\Documents and Setting\Charles DiMaggio\My
Documents\Census\ny_data.txt OBS=26;
```

SAS assumes 256-character record (observation) length or less. You can change this with long record length command (LRECL). For example, here, we tell SAS to read 2,000 characters for each observation:

```
INFILE c:\myrawdata\whatever.dat LRECL = 2000;
```

3.10 Data Input Miscellany

Here are a couple of (somewhat) random points about data input that sometimes come up:

- SAS will automatically read in decimal points if they are in the data. The only reason to specify decimal points is if they are implied but not provided in the data.
- In an INPUT statement, a dollar symbol ($) following a variable name left aligns the character variable, removing any leading blanks. Using the key $CHAR will preserve leading blank values.
- Don't forget the dot after the informat. If you do, SAS will misinterpret it. For example, $3 would be read as a column style input of size one at the 3rd space. You can mix input styles in the same INPUT statement, SAS doesn't care (see pp 36–38 of The Little SAS Book).

3.11 Importing Excel Spreadsheets

An estimated 80% to 90% of all computers use Microsoft Windows or its applications. You will almost invariably have reason to read Microsoft Excel data into SAS. SAS comes with an import wizard that makes the process relatively painless.

Just click through the following steps:

```
File > Import Data >
```

Select Excel from dropdown list and follow the wizards cues.

Note that the import wizard is actually writing code. You will be given the option to save the code. It will look something like this:

```
PROC IMPORT DATAFILE = filename OUT = dataset;
DBMS = identifier REPLACE;
```

If it's a file or file type you will be working on with any frequency, it's a good idea to save the syntax.

Exporting data to a program like Excel is just as simple, by using the Export utility.

Problems

3.1. Reading in Data From the Editor Window

The following data represents blood test results for a series of patients, the first set of numbers is a patient identifier (ID), "Positive" or "Negative" refers to the overall test result (RESULT), the final set of numbers is the absolute test value (VALUE):

```
203769Positive486
201948Positive400
202085Positive364
201755Positive416
202092Positive373
202087Positive657
201358Negative341
201429Positive448
201549Negative320
202741Positive391
201627Positive532
202004Negative268
202052Negative334
203531Positive573
204366Negative348
204042Negative252
```

- Write the syntax to create a data file called "results" in your SAS work library that includes the variables ID, Result, and Level. (You can cut and paste the data from the text file called exercise3_1.txt)
- Print out the data set using PROC PRINT. (You may have to use SAS help or search online for full details on how to use PROC PRINT.)

3.2. Reading in Data from and External Source

- Using the information in the "sparcs layout" text file, create syntax to read a SAS data file into your work library called "sparcs1" as follows:

  - Specify the location of the SPARCS 04 data file on your computer. Include the following variables: DATE, AGE, COUNTY, ZIP CODE, SEX, RACE, ETHNICITY, PRIMARY DIAGNOSIS, and DISPOSITION.
  - Specify a long record length of 450 characters and limit the syntax to read in only the first 100 observations.

- Print the data set out to your output window.
- Use the "Export" utility under the "File" menu to save a comma-delimited (.csv) version of the file on your desktop.

3.3. Creating a SAS Library

Using the libname command from the first line of the intro-sparcs-ami.sas syntax file, create a libref called desk that points to your desktop. (Hint: Right click a file on your desktop and look under the general tab of properties to determine the correct path.)

Chapter 4
Preliminary Procedures

Abstract In this section we'll spend some time looking at simple SAS procedures that you will use frequently. These utility procedures allow you to print, format, and output data in a variety of ways.

4.1 PROC PRINT

PROC PRINT is a useful way to list variables, as well as a (simplistic) tool to get totals and subtotals. In its simplest form, you invoke PROC PRINT, specify your data set, and run the procedure:

```
proc print data=your.data;
run;
```

You can turbocharge PROC PRINT with a number of options:

- A *VAR* statement allows you to select variables to be printed and defines the order of variables.
- *NOOBS* suppresses the observation numbers and in combination with an *ID* statement allows you to specify a more informative variable to identify the observations.
- A *SUM* statement allows you to calculate simple column totals.
- A *WHERE* statement creates subsets by using a comparison or logical operators and functions. Comparisons terms in SAS include EQ (equal), NE (not equal), GT (greater than), LT (less than), GE (greater than or equal to), LE (less than or equal to), and IN (referring to a term contained in the following parentheses, e.g., WHERE DX IN ("AMI," "Pneumonia," "UTI")). Logical operators include the terms AND or NOT. There are also some special operators such as BETWEEN-AND or CONTAINS that can be helpful. We will revisit these operators when we discuss sub-setting statements.

As an example, the following screen shot demonstrates a PRINT command for a data set named "fitness" (Fig. 4.1). We are printing out only the first five

C. DiMaggio, *SAS for Epidemiologists: Applications and Methods*,
DOI 10.1007/978-1-4614-4854-9_4,

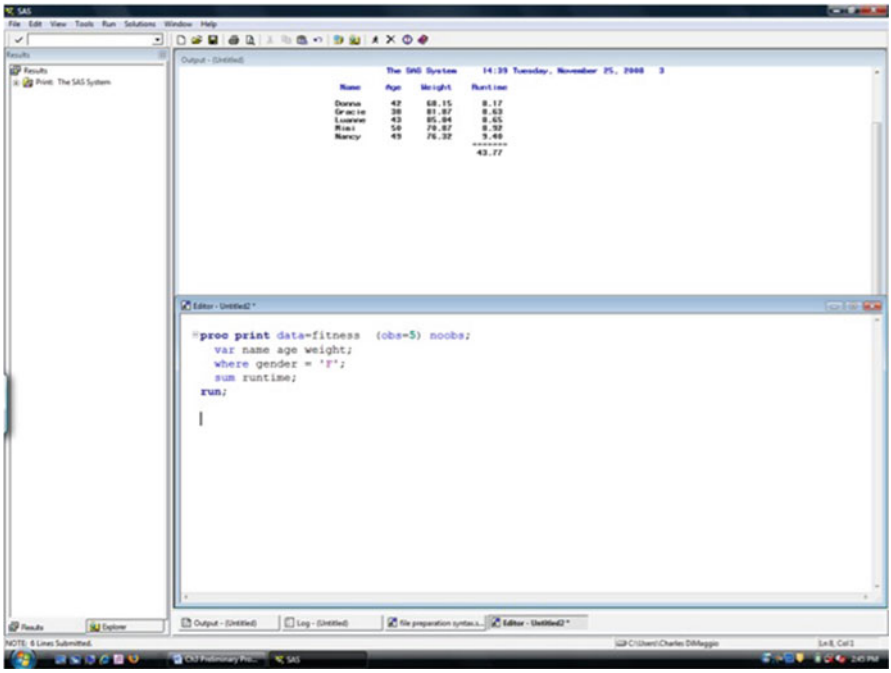


Fig. 4.1 PROC PRINT

observations (note that OBS=*n* has to be in parentheses) and are suppressing the default SAS observation numbers. We could have specified a specific ID variable, but instead, I've listed a name variable as the first of three variables I want SAS to print. I've also restricted the printout to only females and asked for a total of a numeric variable called runtime. Looking back on this paragraph, you can appreciate somewhat the relative brevity and elegance of the syntax.

REALLY. LOOK AT YOUR DATA.

Get in the habit of printing out the first couple of lines of a data set after you create it. Use OBS=*n* to limit the number of observations.

4.2 PROC SORT

After PROC PRINT, PROC SORT is probably among the most useful and frequently used utility procedures. It's used to sequence observations before merging files and grouping observations for subset analyses. For example, PROC PRINT won't sort

observations. If you need subgroups of observations for your analyses, you must use PROC SORT before running PROC PRINT. You can sort an existing file, or you can sort data and create a new file.

Note that PROC SORT by itself doesn't produce any output, although you will receive feedback on the procedure in the log window.[1]

At its most basic, a PROC SORT statement consists of invoking PROC SORT, specifying the DATA set, and telling SAS what variable BY which to sort. The BY variable must be amenable to some sort of logical ordering, either numerically or alphabetically. You can create a new, sorted file using the OUT option (OUT is an option in many procedures) or you can just sort the existing file.

The SAS default is to sort the observations in ascending order of the BY variable. You can, though, choose DESCENDING as an option in front of the BY variable. You can sort by more than one variable, but if you want them *all* in descending order, you need to specify DESCENDING for each variable.

Once you've sorted a data set, you can use SUM as part of PROC PRINT to obtain subtotals.

If we were, for example, to run the following syntax:

```
proc sort data=fitness out= sortdat;
        /*output data set is in the work library */
by gender; /* will be sorted by gender ascending */
run;

proc print data=sortdat;
    /* note using the output sorted data set */
by gender; /* variable by which we sorted */
sum runtime; /* request sum of numeric variable */
run;
```

we would get the following output (Fig. 4.2).

SORT and PRINT is not a very powerful way to obtain subtotals, and you will not likely have much use for it for any but the smallest data sets. PROC SORT, though, is a very useful procedure, and you will likely use it extensively for merging files.

SOME TRICKS

As mentioned earlier, PROC PRINT is not limited to simple BY statements. There are special operators in SAS that allow you to do some pretty cool things. For example, using CONTAINS will return observations that exactly contain a certain character or numeric result. And there are more flexible operators. "$= *$" gives text that sounds like some character string (e.g., where *name* $= *$"*smith*";). NULL or IS MISSING gives observations where the thing is missing (e.g., where name is missing or where name is null).

All these operators work for creating data sets from existing data sets, which we will soon explore.

[1] It is for these procedure that *do not* produce output that checking your log is critically important.

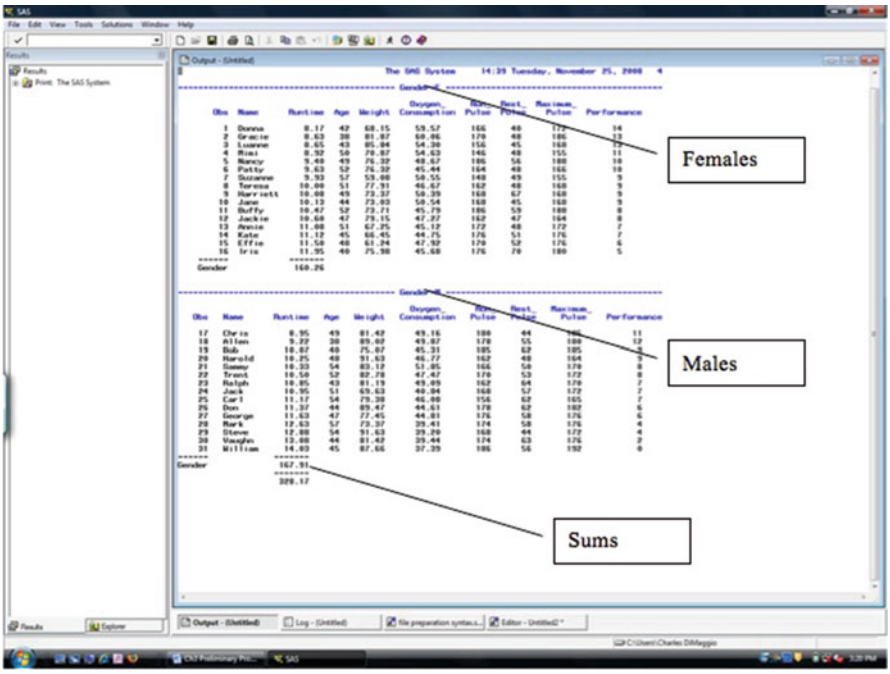


Fig. 4.2 PROC SORT

4.3 Enhancing Output: Titles and Footnotes

Titles and footnotes provide you with an easy way to provide simple descriptions of your output. They can also be used to enhance your output for presentation purposes.

As you might expect, titles will appear at the top of your output, and footnotes will appear at the bottom. You can have up to ten lines of each. It's a good habit to give each of your analyses a title. When you are doing a lot of analyses, this helps keep your output organized and understandable.

You include a title or footnote command as part of your current procedure by writing the word TITLE followed by the text you want to appear enclosed in quotes. The command is, as ever, completed with a semicolon:

```
TITLE This is my title;
```

To have more than one line of text as your title, designate each line with *TITLEn*, where n is the line number:

```
TITLE First Line of Your Title;
TITLE2 Second Line of Your Title;
TITLE3 Third Line of Your Title;
```

The rules are the same for footnotes.

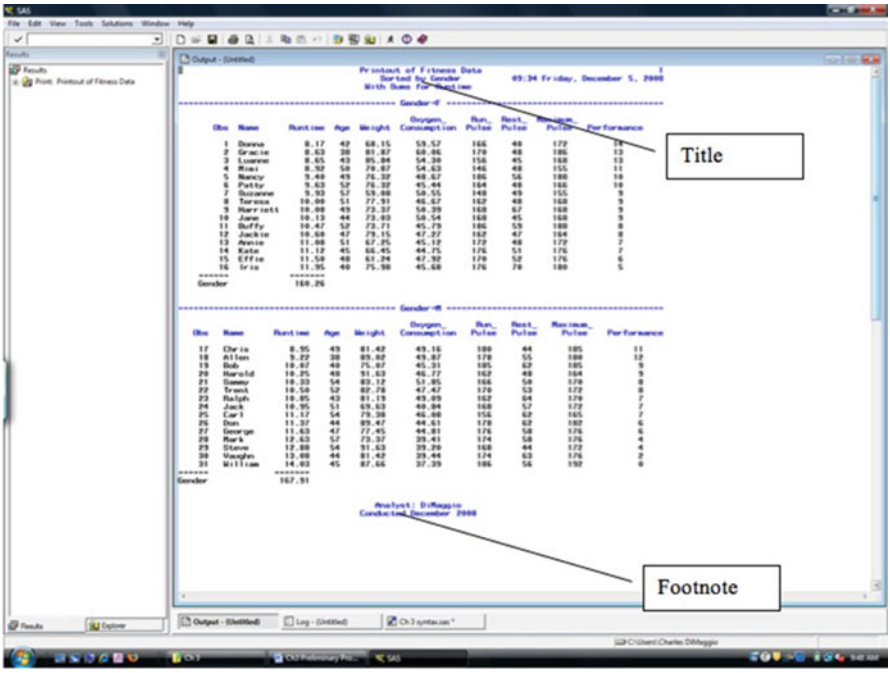


Fig. 4.3 Titles and footnotes

If we were to run the following syntax:

```
proc print data=sortdat; /* using the sorted data set */
title 'Printout of Fitness Data';
title2 'Sorted by Gender';
title3 'With Sums for Runtime';
footnote 'Analyst: DiMaggio';
footnote2'Conducted December 2008';
by gender; /* variable by which we sorted */
sum runtime; /* request sum of numeric variable */
run;
```

You would get the following output (Fig. 4.3).

NEW PAGES

SAS will continue using the same titles and footnotes from the most recent procedure that included a TITLE or FOOTNOTE statement. You need to cancel previous ones by including a blank title and/or footnote statement or write new ones to avoid confusion.

Also, titles and footnotes are hierarchical: issuing a new number *n* title or footnote replaces that one and cancels all subsequent ones.

4.4 LABELS

When you are working with someone else's data, internal SAS variable names can sometimes be inscrutable. For example, consider the following input statement for New York State Medicaid data:

```
input
@1      id              $CHAR20.
@193    ip_dx1          $CHAR6.
@199    ip_dx2          $CHAR6.
@255    cptsx1          $CHAR2.
@309    delivery        1.
@310    ip_days         4.
@314    ip_stat         $CHAR2.;
```

A variable name like id might make sense as identification, but what are we to make of variables like cptsx1? A *SAS LABEL* statement, included as part of the data step, assigns a plain English name to SAS variables. The syntax is straightforward:

```
LABEL
var1=name for variable 1
var2=name for variable 2
var3=name for variable 3
```

So, for the above data statement, a label statement would look like this and would follow the input statement:

```
label
id = 'ELIGIBLE IDENTIFICATION NUMBER'
ip_dx1 ='INPATIENT PRINCIPAL DIAGNOSIS CODE'
ip_dx2 = 'INPATIENT DIAGNOSIS CODE-2'
cptsx1 = 'PROCEDURE CODING SYSTEM CODE'
delivery = 'RECIPIENT DELIVERY CODE'
ip_days = 'MEDICAID COVERED INPATIENT DAYS'
ip_stat = 'INPATIENT STATUS CODE';
```

Now, if you were, for example, to print out a table of diagnoses, the column of diagnoses on your output would be labeled "INPATIENT PRINCIPAL DIAGNOSIS CODE," rather than

```
``ip_dx1''
```

Label names can be up to 256 characters long and are not restricted by SAS variable name rules like not beginning with a number, slash, asterix. etc. Note that the label replaces the variable's name in SAS output but that when referring to the variable in procedures or data steps, you must still use the internal SAS variable name.

Labels can be assigned as part of a DATA step, as in the example above, or they can be assigned as part of a PROC. There is a very important distinction between the two approaches. When you assign labels as part of a DATA step, they are

permanently attached to the variables. When you assign labels during a PROC, they will apply for that SAS session only, i.e., you will need to re-apply them the next time you start up SAS.

NEW LABELS

Once you've defined your labels, for most procedures, they will automatically be invoked. For some procedures, though, you have to specify that you want to use labels. This is the case for PROC PRINT, for example,

```
proc print data = your.data noobs label;
```

4.5 PROC FORMAT and FORMAT

As described in the section above, a LABEL statement is applied to internal SAS variable names to give them plain English or more informative names. By contrast, FORMATS are applied to the internal SAS values that a variable can take. For example, say your data set has a variable called "gender" which is a numeric variable coded "1" for females and "2" for males. (This is a common situation with secondary data sets.) You want to assign the word "Female" to the number 1 and "male" to the number 2 not only because it will make more sense when sharing output with colleagues, but so you don't need to refer to a coding reference every time you return to this data set.

It is essentially a two-step process. You use *PROC FORMAT* to create a format for the variable "gender." (This format which you name and define will live somewhere in the inner recesses of the SAS system.) You then apply this format by using a *FORMAT* statement in a procedure.

Similar to the INFORMATS we discussed in Chap. 2, SAS FORMATS have the form

```
<$>format_namew.<d>
```

where (as with INFORMATS) $ is an optional indicator for a character variable, *w* is a required width definition, the dot is a required delimiter, and *d*, the number after the dot, is the number of decimal points if any.

SAS ships with a number of predefined standard formats. For example, *DOLLAR10.2* would display the numeric variable "11500" as "$11,500.00." The predefined SAS FORMAT *$UPCASEw*. will convert your character variable to uppercase. As with INFORMATS, lists of predefined SAS FORMATS are available in the documentation (hard to find) and on the Internet (easier to find). And also, as with INFORMATS, I find Delwhiche and Slaughter's "The Little SAS Book" an excellent resource in this regard.[2]

[2]pp. 100–101 in the 2nd edition and pp. 110–111 in the 3rd edition.

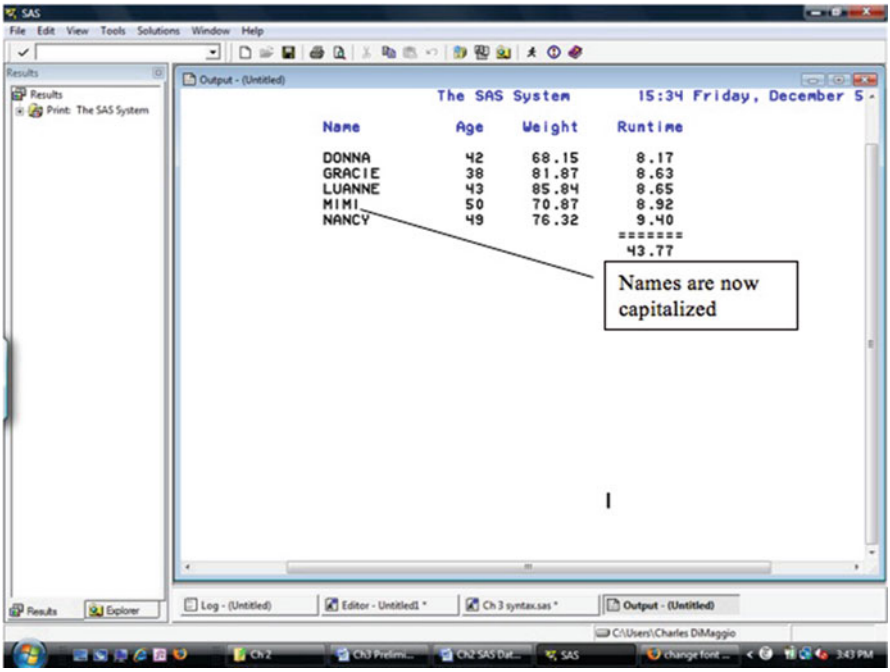


Fig. 4.4 Formatting a variable

If you find a predefined SAS format that fits your needs, you just include it as part of your procedure. Running this syntax will result in the following output (Fig. 4.4).

```
proc print data=fitness (obs=5) noobs;
var name age weight;
where gender = 'F';
sum runtime;
format name $UPCASE10.;
run;
```

Note that, as with LABEL, FORMAT does not actually change the internal source variable, just the way it is displayed.

A nice thing about FORMAT is that (unlike INFORMAT) you are not limited to what SAS has created for you. You can define your own FORMAT using *PROC* FORMAT. We will see that this can be put to good advantage.

The syntax for PROC FORMAT looks like this:

```
proc format;
value format_name
   1='formatted_output_1'
   2 ='formatted_ouput_2';
run;
```

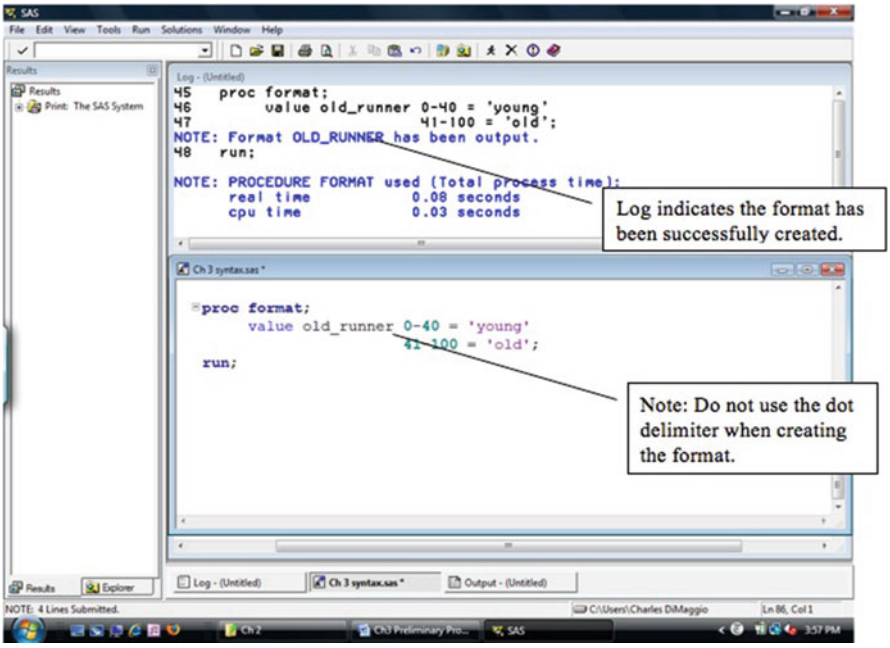


Fig. 4.5 Creating your own format with PROC FORMAT

You begin by invoking PROC FORMAT. Notice that you are not referencing any particular data set. The FORMAT you create will exist in a special place SAS has reserved for such things. The next line of syntax consists of the word "value" then a name for your format. The name cannot be longer than 32 characters and cannot begin with special characters like * or /. If you are creating a few formats for the same variable, you may be tempted to end your format name with a number. Don't. SAS will think you are delimiting a width for the format. Also, notice that when creating a format with PROC FORMAT, we do not end the format name with a dot delimiter. The delimiter is used when the format in a procedure is invoked.

In the following syntax, we are using PROC FORMAT to create a format that we will apply to the ages of runners. We specify a numeric range of 0 to 40 that we will consider "young" and a range of 41–100 that we will consider "old" (Fig. 4.5).

Now that we have created this new format, we will apply it to the data we have been printing. Note that the relevant line of syntax is

```
format age old_runner.;
```

and that *we now include the delimiting dot*. You will notice that when you put a dot after a word in the Enhanced Editor, SAS colors it a pleasant aqua.

```
proc print data=fitness (obs=5) noobs;
var name age weight;
where gender = 'F';
```

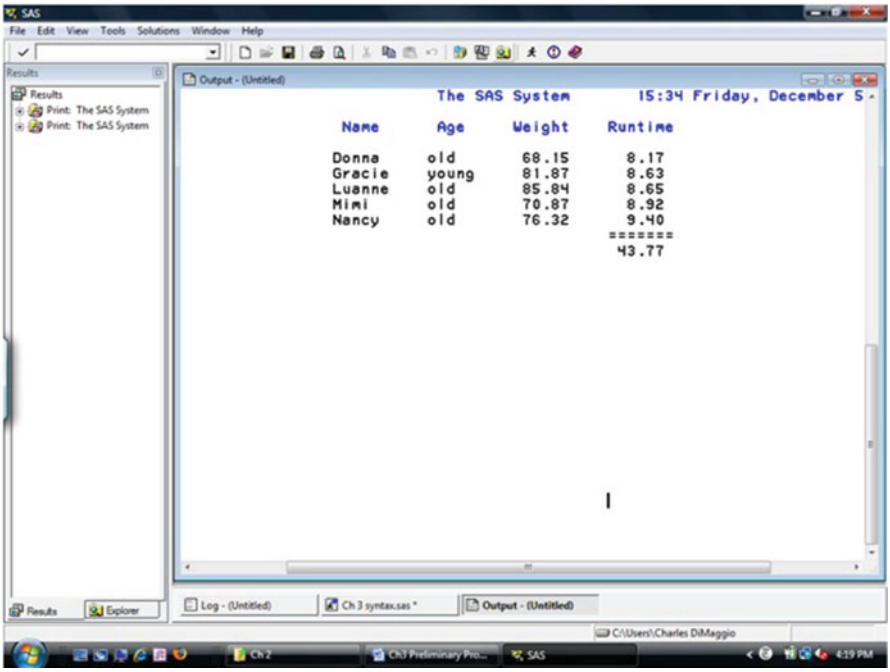


Fig. 4.6 Using PROC FORMAT to categorize a variable

```
sum runtime;
format age old_runner.;
run;
```

The above syntax will give you this output (Fig. 4.6).

Note that we assigned a name to a range of numeric values. In the world of science, where folks tend to be either splitters (breaking things down to their smallest constituent parts) or lumpers (combining things to find patterns), epidemiologists tend to be lumpers. Formatting numeric ranges is an easy way to lump things together and perhaps begin to see some patterns in your data.

Our example involved formatting a numeric variable to display character values. Look at the syntax. You will see that the format values are enclosed in quotes to indicate that they are character values. If we wanted to format this numeric variable to display a number instead, say 1 for young and 2 for old, they would not be in quotes. The name of the format itself (old-runner) could basically have been anything.

If the original variable to which we were applying a format were a character variable, the name of the format we create would have to start with a $ sign. In the sample syntax below, we are formatting a character variable for a school grade that has SAS internal values of A, B, C, and D. Notice that both the internal values and the formatted values are enclosed in quotes and that the name of the format begins with a $ sign.

```
proc format;
value $grade
'A'='good'
'B'='fair'
'C'='poor'
'U'='see
instructor';
run;
```

When you apply this format as part of a procedure, you will also need to include the $ sign:

```
proc print data=your.dataset;
format final_exam $grade.;
      /* again note need for dot when applying*/
run;
```

Much like a LABEL statement, when you define a FORMAT as part of a PROC, it will apply for that SAS session only. The next time you start up SAS, you will have to redefine the FORMAT. When you define a FORMAT as part of a DATA step, the FORMAT will be permanently associated with that data set. Unfortunately, unlike a LABEL statement, the FORMAT is not actually attached. SAS knows there is a FORMAT associated with a variable, because it was part of the DATA step, but it doesn't know any of the details about the FORMAT. When you try to run a procedure on a variable that has such a DATA-defined format associated with it, you will get an error message and will still need to reinvoke the FORMAT much as you would if it were defined as part of a PROC.

In fact, the situation may get more complicated. Depending on how the formats were defined, SAS may not even allow you to open the data set.

There are a couple of ways around these difficulties. The most straightforward and easiest (and hence the one I tend to use) is to just include formats as part of the syntax for a procedure. You could alternatively run the FORMAT statement(s) every time you open up the data set. Or you could create something called a format library that permanently stores the format information. You do this with a LIBNAME statement where the libref is the word "library."

PUBLIC LIBRARY

Many publicly available data sets, for example, CDC's Behavioral Risk Factor Surveillance System, come with format libraries that can be difficult to use and apply. This can be frustrating, especially if you just want to run a couple of quick analyses. If you don't properly download, save, and identify the library, you will get numerous error messages, and sometimes the file won't run at all. In this situation, the simplest solution is to include a *NOFMTERR* option when opening the data set.

4.6 ODS

ODS stands for Output Delivery System. It is a SAS tool that allows you to save your output in a variety of different file types including Microsoft Excel, .pdf, .html, and .rtf (rich text format, for MS Word documents). You start ODS by simply typing the letters "ods," then specifying the file type you'd like ODS to create, and then specifying where on your computer you'd like ODS to create that file. I usually specify someplace on my desktop. When you specify the file location, it must be enclosed in quotes and have the appropriate file extension. No RUN statement is necessary for this opening ODS statement. You then write out whatever PROC from which you want output as you normally would, including a RUN statement. You then close ODS.

As an example, let's say that we are out to impress, and the simple SAS output for runners' ages, weights, and run times from our previous example just won't do. HTML is a fairly versatile file type that can be opened by many programs, and SAS provides a nice output style for HTML called "sasweb."

Below, you will see the syntax on the lower half of the screen and the usual SAS output on the upper half of the screen (Fig. 4.7).

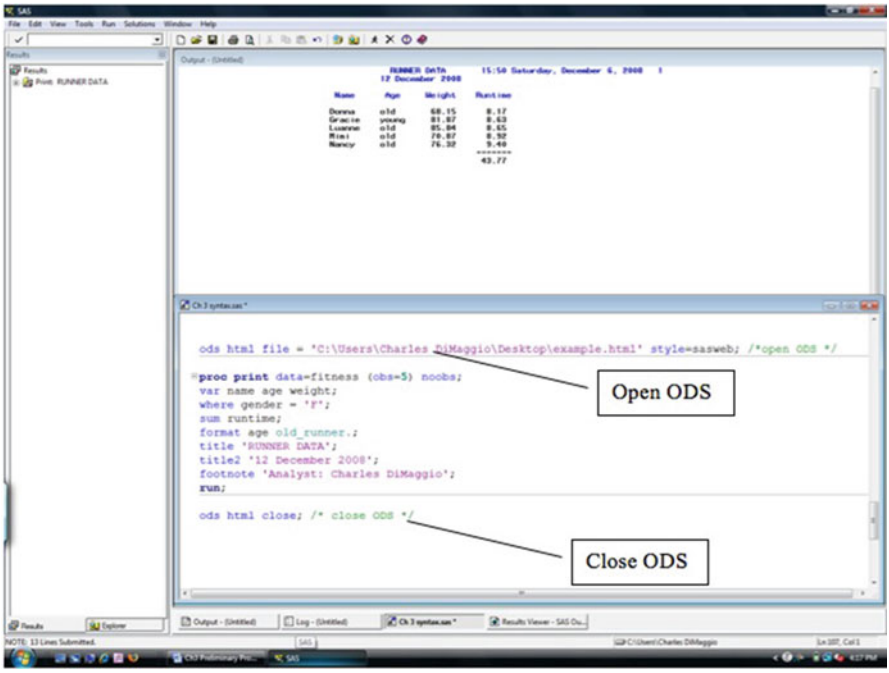


Fig. 4.7 Output delivery system (ODS)

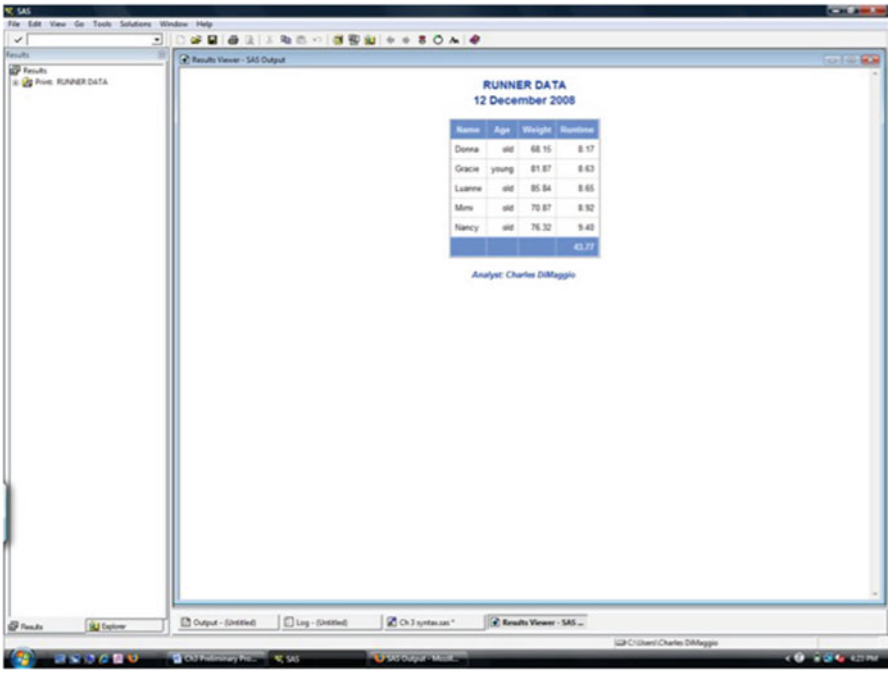


Fig. 4.8 PDF output using ODS

But, by invoking ODS, you get additional output. First, SAS creates a new internal window called Results Viewer that contains your ODS output. The output certainly looks nicer (Fig. 4.8).

More importantly, SAS creates a file with that nice output, on your computer, based on your specifications. The following screen shot is of the HTML document we created, which I opened using my Firefox browser. I could also have opened it with Microsoft Word, which would have made it easy to include it in any report or manuscript in which I might want to include it.

ODS has many tricks up its sleeves, and if you spend much time on SAS, you will surely find use for it in many important ways. For our purposes, knowing how to save your output in different file forms is sufficient. In the above example, if you wanted a pdf file instead of an html file, you would simply have opened the pdf engine with ods and specified a pdf file extension like

```
ods pdf file=c:\temp\example.pdf
```

There is, though, no nice formatting option for a pdf file like sas web, so it might be better to use ods to create the html file then save or print the html as a pdf.

SIMPLE TABLES

You may prefer setting up and formatting tables in a more familiar program like Microsoft Excel or Word. You can use ODS as a neat way to output a table of results directly into Excel so you can use it in a Word document. Open ODS HTML and just put an .xls extension for the file. Make sure you specify a "minimal" style to get rid of any extraneous SAS formats.

```
ods html file = 'C: \table.xls'  style=minimal;
```

A QUICK REVIEW

- LABEL: specify up to a 256-character label for each variable in an existing data set
- FORMAT: define how to display variable values in an existing data set (if used in DATA step, permanently assigns format, if used in PROC step, just for that session)
- PROC FORMAT: to create your own formats
- INFORMAT: to assign a SAS format type to a variable when inputting data
- LIBNAME: used to create a data library name

Problems

4.1. Using PROC PRINT

You are a hospital epidemiologist. The medical director needs some (very) quick information on what group of patients requires the most resources. Not knowing that all you know how to do is PRINT data, she hands you SAS data file exercise4-1. She's not very clear about what's on it, but she thinks it can help you answer her question.

(a) Begin by exploring the data set. Copy the file into a folder on your computer. Write a LIBNAME statement to locate it. Open the file using the SAS Explorer. What kind of file does it appear to be? How many observations are there? How many variables?

(b) Looking at a few variables may help you get a quick sense of where resources are going. You recognize AGE as age of the patient, LOS as length of stay in days, CHARGE1 is how much the stay costs the hospital, and PDX as primary diagnosis. Write syntax to print the variable's age, primary diagnosis, and charge. Limit your output to the first 20 observations. Do not print the observation numbers. Print only those observations where the length of stay is greater than 14 days and request a total for the costs charged to the patient.

(c) What is the total charge for patients whose length of stay was greater than 14 days? What is the total charge for patients whose length of stay was less than 7 days? Based just on this incomplete information, what can you tell the medical director?

4.2. From SAS to Excel

One of your colleagues is interested in the results of your initial work with the exercise4-1 file. He asks if you could create an Excel file for him so he could do some analyses of his own. He also asks if you have a sense of what the average age and gender of patients is.

(a) Create an Excel file of exercise4-1 listing the variables for date of service, age, gender, primary diagnosis, and disposition for the first 150 observations.
(b) Within Excel, calculate the average age. What is the average age for males vs. females?

4.3. Creating and Applying Formats

The medical director decides she wants to make a presentation to the hospital board using your results. She asks if you could spruce up the output you showed her.

(a) Apply the dollar11.2 format to the CHARGE variable and print out the first 20 observations.
(b) Use PROC FORMAT to create a 3-level format for the charge variable named dol-range, with low defined as the lowest charge up to 500,000, medium defined as 500,001 to 1,000,000, and high defined as 1,000,001 to the highest value. Apply this format and print out the first 20 observations.[3]
(c) Create and apply a format to the sex variable that formats F as female and M as male. Apply this format and print out the first 20 observations.

4.4. Using Titles and Labels

As an additional task, you are asked to print up the results of a series of blood tests from a research project.

(a) Open the text file Exercise4-2 and cut and paste the data into your editor window.
(b) Read the data into a SAS data file called "test" in your work library. (Hint: See exercise from previous chapter on reading in data from editor window.)
(c) Title your output "Initial Blood Test Results." Label the variables as follows: ID = "Patient Identifier," RESULT = "Final Result," and VALUE = "Assay Level." Print the output.

[3]The CHARGE variable is actually more complicated that with which you are working. It is a 10-digit SAS numeric variable which includes cents and right filled with zeros, so these numbers are not the actual charges. Don't worry about this.

Chapter 5
Manipulating Data

Abstract In this chapter we spend some time discussing how to use DATA and SET statements to create new SAS variables, new files out of old files and different ways to combine and merge data sets.

As we've discussed, there are two basic types of SAS statements: PROC and DATA. As important as PROC steps are to plumb the depths of your data, much of the hard work of epidemiological data analysis in SAS occurs during DATA steps. You will use DATA steps not only to read in and create new data sets but, perhaps as importantly, to clean, subset and merge data sets, as well as to create new variables. It is during DATA steps that you take raw observations and manipulate them into forms amenable to analysis. And it is frequently the DATA steps that require most of the real thought that ensure the reliability and validity of your work. In fact, after wrestling with DATA steps, the PROC steps seem almost trivial. The good news is that SAS is extraordinarily effective and flexible in manipulating even the largest and most complex data sets. It is this capability that distinguishes it from many other data analysis tools.

5.1 The SET Statement

Once you've read in your raw data into a SAS data set, you can create other SAS data sets based on it. The key to creating and manipulating SAS data sets is the SET statement. In general, when manipulating existing SAS data sets, the SET statement takes the place of the INFILE and INPUT statements we've discussed.

The general form of a DATA step with a SET statement is

```
DATA    newDataset;
SET     oldDataset;
        additional statements manipulating the existing data
          set;
RUN;
```

C. DiMaggio, *SAS for Epidemiologists: Applications and Methods*,
DOI 10.1007/978-1-4614-4854-9_5,

READY, SET...

As you can see, a SET statement is part of a DATA step that creates a new data set. If you name your new data set with the same name as the existing data set you will irretrievably replace the old data set.

If not used with appropriate care and respect, SET statements can destroy an existing data set. Proceed with caution. Always use a new name.

5.2 Using SET to Define and Create New Variables

We spend a lot of time in epidemiology counting, adding up, subtracting, and dividing variables. All the concepts you may have learned about risk differences, rate ratios, and odds can only be applied when you have the appropriate variables on which to perform these kinds of operations. SAS has great capabilities in this regard.

As described above, you first name or specify a new data set using a DATA step, then you use a SET statement to identify an existing data set you will be manipulating to create that new data set. You can use the SET statement to create new variables in the existing data set or add, multiply, or apply functions to existing variables in the existing data set.

The basic format to create a new variable is (simply enough)

```
newVariable = expression;
```

So, for example,

```
var = 10;
```

creates a new numeric variable for every observation called var that is a constant with value 10.

```
var=ten
```

creates a new character variable.

```
var2= var1*10;
```

creates a new variable named var2 that is the product of an existing variable, var1, multiplied by 10.

5.2.1 Operations

Mathematical or logical operations in SAS tend to be familiar and often have character equivalents:

```
+   add
-   subtract
```

```
* multiply
/  divide
** exponentiate
= or EQ  equal
> or GT  greater than
<= or LE  less than or equal
>= or GE greater than or equal
```

5.2.2 *Functions*

In addition to straightforward operations, SAS comes with a plethora of canned routines, called *functions*, that are sort of automated procedures. They come in the basic form

```
functionName(argument, argument )
```

So, for example,

```
newVar = INT(oldVar);
```

returns the integer portion of the value in the parenthesis, and

```
newDateVar = DAY(oldDateVar)
```

returns the day names of dates.[1] You can even put another function as the argument of a function.

There are functions that will help you convert character variables to numeric variables and vice versa, functions to string two or more variables together, and functions to strip out pieces of variables. In fact, if you can think of something you'd like to do to or with a variable, whether it is numeric or character, there's a good chance SAS will have a function that addresses it.

You can unearth a full list of SAS functions and their associated procedures in SAS help by following these steps through the menu:[2]

```
help -> SAS help and documentation -> SAS products - Base
    SAS ->
SAS Language dictionary -> dictionary of language elements ->
SAS data set options
```

THERE'S A FUNCTION FOR THAT

SAS functions aim to please. So much so that they may return a result even if you feed them a variable with missing or nonsensical values. Before using a function on a variable, perform some simple procedures like MEANS or FREQ on it, and check your results.

[1] In this case, the argument oldDateVar must be a valid SAS date variable.

[2] Again, SAS help is excellent, if not immediately reachable. Delwiche and Slaughters' 2nd edition lists some popular functions in Sects. 3.2 and 3.3.

5.2.3 *Example: Deaths Following the Terrorist Attacks of September 11, 2001*

Let's look at an example of using a DATA-SET sequence of statements to manipulate variables in a data set. The file ch5demo1 is a New York City Office of the Chief Medical Examiner data set of deaths related to the terrorist attacks of September 11, 2001, grouped by zip code. Let's explore the file. You should, by now, be familiar with how to do this.[3]

We see that males and females are listed separately (Fig. 5.1). We'd like a variable for the total number of deaths in a zip code. The following syntax accomplishes that:

```
data nyc_deaths;
set ch5.ch5demo1;
tot_911_deaths = female_911_deaths + male_911_deaths;
run; /* note there is no output, check log window */
```

A DATA-SET run like this will not produce any results in the output window. It is, though, crucially important to check your log window to make sure the process ran correctly (Fig. 5.2).

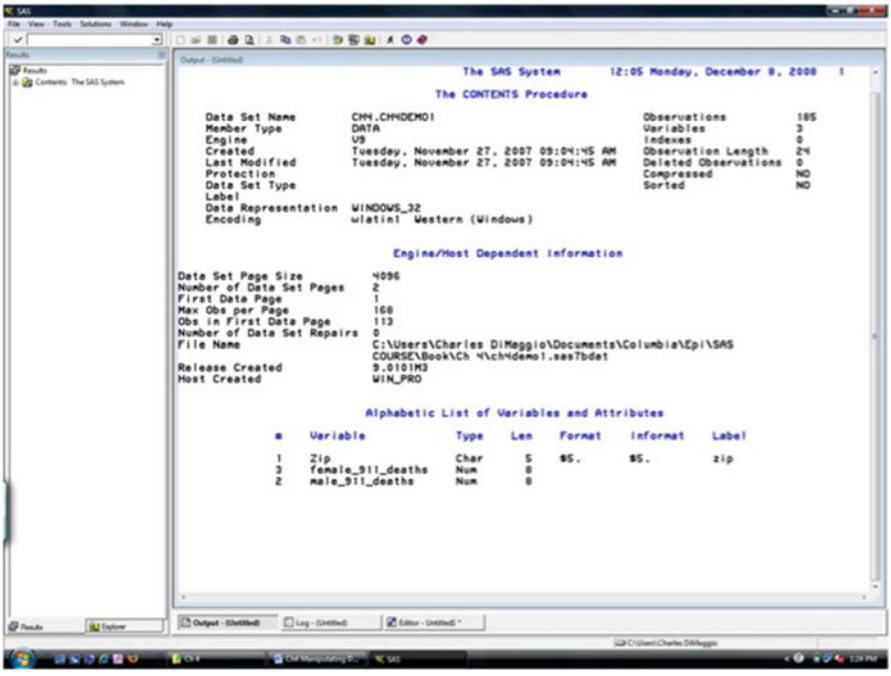


Fig. 5.1 Contents of the September 11, 2001, deaths file

[3]Hint: PROC CONTENTS.

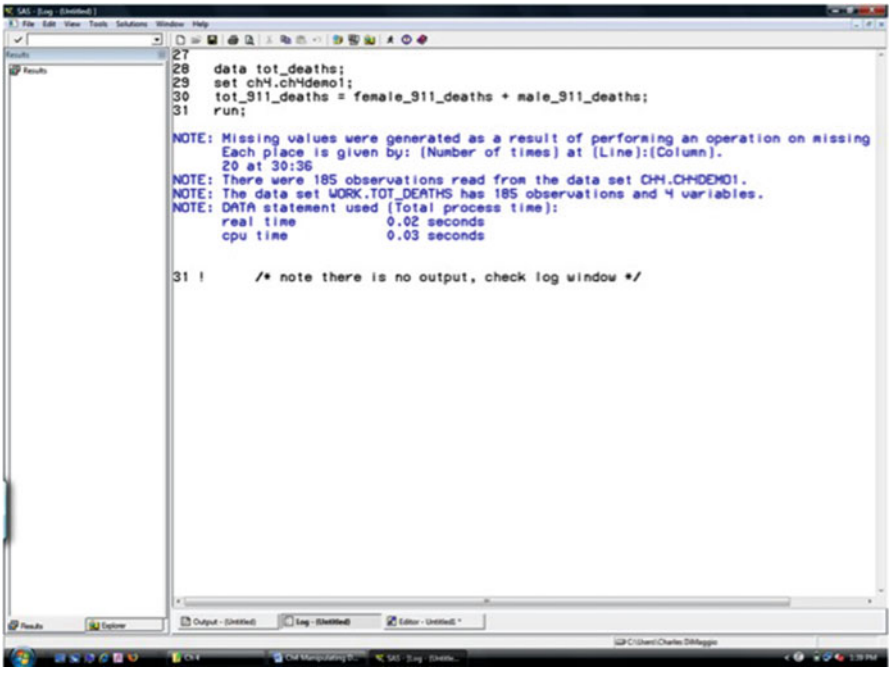


Fig. 5.2 Log file for data manipulation

There is no dreaded red font, so we know that the procedure ran. But, there are a number of notes, the most important of which is that "missing values were generated as a result of performing an operation on missing values.". This occurred 20 times at positions 30 and 36.[4] Let's take a look at the file. We could print up the observations, but in this case, let's drill down to it using SAS's Explorer window (Fig. 5.3).[5]

We see that when a zip code tabulation area did not have an entry for both male and female deaths, SAS (appropriately) returned a missing value for total deaths. We also see that there were no observations where there was a male or female death in a zip code and a missing observation for the opposite gender. The coder placed zeros when that occurred. Keep in mind that SAS may or may not return a result when there is a missing value for one of the arguments depending on the operand or function.

[4]Why did this occur? Some zip code tabulation areas may not have had any deaths assigned to them, or they may have been male but not female deaths, or vice versa. The position of our male and female death numbers is 30 and 36, respectively.

[5]We will find the file in the work library.

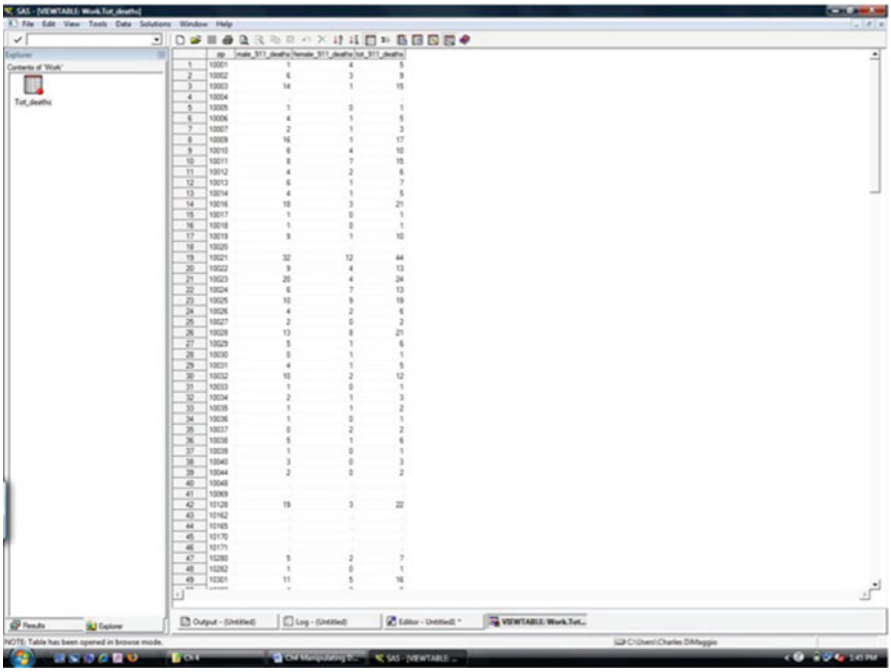


Fig. 5.3 Reviewing the new data file

5.3 Adding (Concatenating) Data Sets

Concatenation refers to adding or stacking one data set on to another. We will again use SET as part of a DATA step to concatenate or stack data sets by using a SET statement to append data sets to each other. The basic syntax looks like

```
DATA newDataset
SET oldDataset1 oldDataset2
```

As before, the DATA statement names the new data set to be created. When concatenating data sets, the SET statement consists of a list of the existing data sets you want added to each other. The operation simply adds the data sets to each other observation- or row-wise. So, if you add one data set with 5 observations (e.g., 5 patients) and 3 variables to another with 5 different observations (i.e., 5 other patients) and 3 *different* variables, the resulting new data set will have 10 observations and 6 variables. If both sets have the identically same variables (i.e., same variable name, same variable type, same format, etc.), the resulting data set will again have 10 observations but will have only 3 variables.

Concatenation does not *merge* observations. One set of observations is simply *added* to or "stacked on top" of another. If, for example, you added data set A

to the exact same data set A, the new data set will contain the same number of variables with the observations duplicated. As with previous SET statements, during concatenation you can manipulate existing variables to create new variables.

5.3.1 Concatenating the September 11 Data Set

Continuing with our 9/11 terrorist deaths example, you soon realize that deaths were not limited to New York City residents. You obtain a separate file of deaths for non-New York City deaths. Ch4demo2 contains these data. Your next step is to add the New York City file to the non-New York City file:

```
data tot_deaths; /* creating new file in work library */
set nyc_deaths ch5.ch5demo2; /* adding file to existing non_NYC
  file */
run;
```

You receive something like the following in your log window:

```
11
12   data tot_death; /* creating new file in work library */
13   set nyc_deaths ch5.ch5demo2; /* adding file to existing
        non_NYC file */
14   run;

NOTE: There were 185 observations read from the data set
         WORK.NYC_DEATHS.
NOTE: There were 1421 observations read from the data set
         CH5.CH5DEMO2.
NOTE: The data set WORK.TOT_DEATH has 1606 observations and 4
         variables.
NOTE: DATA statement used (Total process time):
      real time           0.01 seconds
      cpu time            0.01 seconds
```

Note that the 1,606 observations in the new data set are simply the sum of the 185 observations in the New York City data set plus the 1,421 observations in the non-New York City data set.

5.4 Merging Data Sets Using MERGE – BY

Often when working with secondary data sets you will find that your file does not contain all the variables you would have included had you designed the study yourself. So, for example, you might want to know some information on the socioeconomic status of individuals. If you don't have that individual-level data, you may choose to use income information from the area in which a person resides. If you know the person's zip code, you could look up the median household income for

that area from census records. You can then merge a file of zip code tabulation areas and their median household income statistics to your file of individual observations. SAS makes such operations easy. (Perhaps too easy, as we shall see.)

To match and merge data sets based on some key or identifying variables in SAS, simply substitute the word *MERGE* for the word SET and include a *BY* statement to identify the key variable by which you wish to connect the observations:

```
DATA  newDataset;
MERGE oldDataset1 oldDataset2;
BY idVariable;
```

where idVariable is a variable present in both data sets that uniquely identify an observation.

This powerful procedure differs fundamentally from simply concatenating data files using a SET statement. If you have, say, 100 observations in one file and merge them on a one-to-one basis with another file of 100 observations, your resulting file will have 100 observations.[6]

5.4.1 SORT Before You MERGE

Whereas you can add or concatenate two files together as they are, to merge files, they have to be sorted by the key variable by which you wish to match them up. That bears repeating:

To MERGE, first SORT

As presented in Chap. 3, PROC SORT is used to sort a data set.

You can merge more than 2 data sets. You can do one-to-one (BY variable not repeated in either data set), one-to-many, or many-to-one (BY variable unique in one data set and repeats in the other) merges. SAS doesn't care, and the syntax is the same.

CAREFUL. MERGE AHEAD.

Merging is a powerful procedure. It should at all times be used with caution. In addition to the usual warnings about looking at your log file for errors, you should carefully review the resulting data set. If, for example, two data sets have the same-named variable, one of them will be discarded. If you do something like neglect to include a BY variable, there's no telling what the resulting data set will be.

[6] How many observations would you have if you concatenated using a SET statement?

5.4.2 *Merging the 9/11 Data Set*

As an example of merging, we return to our 9/11 data set. We know as good epidemiologists that a simple count of events is not nearly as informative as a rate. We have the number of deaths in a zip code tabulation area, but we need some estimate of the number of individuals at risk. A simple first step would be to base a rate on the number of individuals living in a zip code tabulation area. This information is available from census figures and is contained in the file Ch5demo3. Take a moment to explore it.

You will find that there are a number of helpful variables in this file, but first, we need to merge it to our existing totDeaths file. We begin by identifying a variable by which to merge the data sets. The variable has to be present in both files and defined the same (numeric, character, date) in both files. Zip is an obvious choice. The following syntax should merge the files based on zip code:

```
data pop_deaths;
       merge tot_deaths ch5.ch5demo3;
       by zip;
run;
```

Oops. Running that syntax will return the following error in our log window:

```
ERROR: BY variables are not properly sorted on data set
         WORK.TOT_DEATHS.
Zip=11697 male_911_deaths=6 female_911_deaths=1
         tot_911_deaths=7
POPBLACK=5 POPASIAN=13 POP_HISP=52 POP_2024=190 POP_2534=407
         POP_3544=605
POP_4554=616 MHI=58491 POP_TOT=4226 FIRST.Zip=1 LAST.Zip=1
         _ERROR_=1 _N_=358
NOTE: The SAS System stopped processing this step because of
         errors.
NOTE: There were 186 observations read from the data set
         WORK.TOT_DEATHS.
NOTE: There were 359 observations read from the data set
         CH4.CH4DEMO3.
WARNING: The data set WORK.POP_DEATHS may be incomplete.
         When this step was stopped there were 357 observations
         and 13 variables.
```

We forgot to sort by our "BY" variable, zip. Remembering PROC SORT from Chap. 3, we write the following syntax:

```
proc sort data=tot_deaths;
     by zip;
proc sort data=ch5.ch5demo3;
     by zip;
run;
```

After sorting, we run the merge statement again. This time, for good measure, let's just keep the two variables in which we are interested:

```
data pop_deaths(keep= zip tot_911_deaths mhi);
merge tot_deaths ch4.ch4demo3;
by zip;
run;
```

This runs smoothly, and the log does not return any error messages.

TWIST AND MERGE

There's a neat little twist you can use when merging data sets. The "IN=" option when used with MERGE identifies the data set an observation comes from.

It creates a temporary variable called "indata1" that indicates whether the variable came from a data set. 0 means a data set didn't contribute to the current observation, 1 means it did. This is cool enough in its own right, but used with an IF statement (which we'll talk more about shortly) we can use it to eliminate non-matching observations. The following syntax creates a data set with only merged observations from the data set "old1".

```
DATA new; MERGE old1
(IN=selected) old2;
BY var;
IF selected=1;
run;
```

As noted above, a rigid requirement for merging files is that the BY variables must match exactly, i.e., type (numeric vs. character), name, and length. You may have to spend a fair amount of time preparing data sets for merging.

If you need to convert a character variable to a numeric variable use INPUT, which we first encountered when we discussed reading data into a SAS data set, as part of a DATA-SET statement to create a new variable. As with our previous encounter with INPUT, you will have to specify a numeric INFORMAT that best describes how to read the character data value into the new numeric variable.

If you need to convert a numeric variable to a character variable, you will again use a DATA-SET statement to create a new variable, but this time, you don't use INPUT. Rather, initialize the new variable as a character variable by using the dollar sign ($) and define its length. Then set the new character variable equal to the old numeric variable. For good measure, align it to the left to eliminate all leading blanks.

CONVERTING VARIABLES.

SAS is very strict about BY variables in a merge. They must be the same type, length, etc. You may find yourself in the position of having to manipulate variables so they will merge. The following syntax may come in handy:

1. Convert a character variable to a numeric variable using input()

```
data  new_data; /* create new data set */
set  old_data;
num_var=input(char_var,informat.);
/* e.g. numeric informat 3. */
run;
```

2. Convert a character variable to a numeric variable using put()

```
data new_data;
set old_data;
char_var = put(num\_var, informat.)
/* e.g. character informat $3. */
```

BY ANY OTHER NAME

When creating new data sets, you can rename variables on the fly. Put the RENAME option next to the data set in which you want to rename the variable. If renaming more than one variable, put them each in parentheses:

```
DATA newData:
SET oldData1 oldData2
(RENAME=(old_var=new_var) );
RUN;
```

5.5 Conditional Expressions Using IF-THEN-ELSE

When we discussed formats in the last chapter, I mentioned something about science consisting of lumping things together or splitting things apart and that epidemiologists are very much "lumpers." While partly tongue-in-cheek, we do as epidemiologists tend to group things together and see if there are patterns in the groups or if the groups are associated with each other or some other factor.

We saw in the previous chapter that proc format is an easy way to group variables for descriptive purposes. It does not, however, actually create a new variable on which you can perform more sophisticated statistical procedures. IF-THEN-ELSE statements are used quite logically and intuitively as part of DATA steps to create such new grouped variables based on existing attributes.

As opposed to our approach in Chap. 3, we would simply write

```
IF age < 40 THEN ageGroup = young;
```

Here we create a new character variable called ageGroup based on the existing numeric variable age.

We need to complete the IF-THEN-ELSE statement to create a set of mutually exclusive groups:

```
IF age < 40 THEN age_grp = young;
ELSE IF age GT 40 THEN age_grp = 'old';
ELSE age_grp = ' ';
```

You could now, if you were looking at a dichotomous outcome such as mortality, calculate a familiar epidemiologic measure for age such as an odds ratio.

A couple of points about this syntax are in order:

- You can use mathematical operators (=, >, <, >=,etc.) or comparison operators (EQ, NE, GT, LT, GE).
- IF THEN statements are case sensitive.
- If creating a character variable, it needs to be in quotes.
- Always try to include a concluding ELSE statement. This statement has no IF-THEN attribute. It is more efficient computationally and insures that the categories are exclusive.

Another minor but potentially important point: SAS uses the length of the first variable read into a data set to define the variable. So, in a statement like,

```
if dx=one then diagnosis =flu;
else if dx=two then diagnosis =Urinary Tract Infection;
else diagnosis = ;
```

because the first diagnosis defined is the 3-character variable flu, SAS will define the variable diagnosis as 3 characters long and truncate any following diagnoses. urinary tract infection will be cut off to Uri. In a simple case like this, you could just define the urinary tract infection variable first. It would be better, though, to use a LENGTH statement to define the variable explicitly by preceding the IF-THEN statement with

```
length diagnosis $ 11;
 /*note no period after the length number like with format*/
```

THE IN LIST

Rather than write an IF-THEN statement for each possible value of an existing variable, you can use an *IN* statement to create a look-up list.

An IN statement compares the value of a variable to a list of variables:

```
IF sx IN (cough, ache)
   THEN dx = flu;
```

For a demonstration of the power of IF-THEN-ELSE statements, let's turn to our New York State SPARCS data set. Say, we are interested in the mortality associated with substance use and with child abuse. There is no explicit mortality variable in the data set, but there a disposition variable that includes death as one possible outcome. Similarly, while there is no single, explicit substance use variable, or child abuse variable, we can create one using ICD-9 codes.[7] We need to create 3 new variables based on the existing variables. The following syntax will accomplish that:

```
/* IF-THEN-ELSE Demonstration */

DATA NYCSPARCS; /* name the data set in work directory */

INFILE 'C:\Users\Charlie\Documents\Columbia\Epi\SAS COURSE\Data
Sets\nycsparcs.TXT'
MISSOVER LRECl=452 OBS=15000;
/* tell SAS where file is, that its a long file and to
 read in the 1st 15,000 observations */

INPUT /* input the variables you are interested in */
@18   DATE      yymmn6. /* informat based on SPARCS raw file */
@44   AGE       3.
@71   PDX       $CHAR6.
@331  ECODE     $CHAR6.
@343  DISPO     $CHAR2.
;
RUN;

PROC CONTENTS DATA=NYCSPARCS; /* check your file was read
    in */
RUN;

/*******************MORTALITY ************************/
data nycsparcs;
set nycsparcs;
IF DISPO = '20'  then death=1;
                 else death=0;

/********************** SUBSTANCE ABUSE ****************/

if pdx in /* use ICD9 codes to create diagnoses */
('2910','2911','2912','2913','2914','2915','29181','29189',
'2919','2920','29211','29212','2922','29281','29282','29283',
'29284','29289','2929', '30300','30301','30302','30303',
  '30390',
'30391','30392','30393','30400'
'30401','30402','30403','30410','30411','30412','30413',
  '30420',
'30421','30422','30423','30430','30431','30432','30433',
  '30440',
```

[7] International Classification of Diseases, 9th edition is a widely utilized system that assigns numbers to diseases. It dates back to the work of William Farr in nineteenth-century England.

```
'30441','30442','30443','30450','30451','30452','30453',
  '30460',
'30461','30462','30463','30470','30471','30472','30473',
  '30480',
'30481','30482','30483','30490','30491','30492','30493',
  '30500',
'30501','30502','30503','3051','30520','30521','30522','30523',
'30530','30531','30532','30533','30540','30541','30542',
  '30543',
'30550','30551','30552','30553','30560','30561','30562',
  '30563',
'30570','30571','30572','30573','30580','30581','30582',
  '30583',
'30590','30591','30592','30593')
Then subst_ab=1;
Else subst_ab=0;

/************ CHILD ABUSE **********************************/

if age LE 10 AND ecode in
('E9670','E9671','E9672','E9673','E9674','E9675','E9676',
'E9677','E9678','E9679','E9684','E9040','E9041','E9042',
'V1541','V1542','V1549','V6121')
then child_ab=1;
else child_ab=0;

RUN;
```

Our first step is to check our log for any error messages (Fig. 5.4).

There are no error messages. Next, let's check the contents of this new data file (we included a PROC CONTENTS statement in the syntax) (Fig. 5.5).

So far, so good. Now, let's print out some of the data set to assure ourselves we have the variables we need. We will also request some quick totals. The following syntax will accomplish that:

```
PROC PRINT DATA=NYCSPARCS;
VAR death subst_ab child_ab;
sum death subst_ab child_ab;
run;
```

The bottom of our output will look like this, showing that there were 154 deaths, 3,460 cases of substance misuse, and no cases of child abuse (Fig. 5.6).

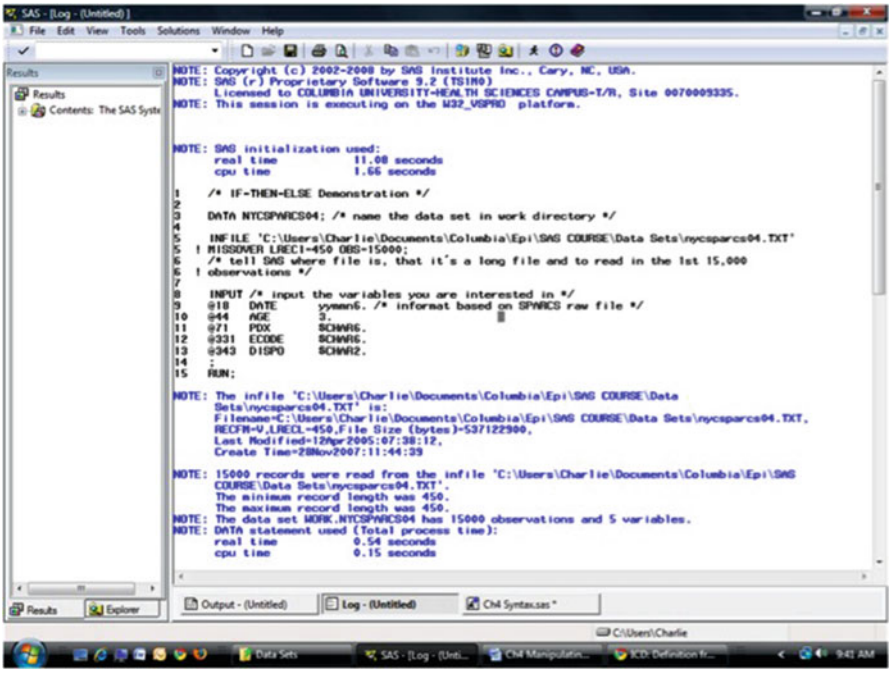


Fig. 5.4 Log window for data created with IF-THEN-ELSE variables

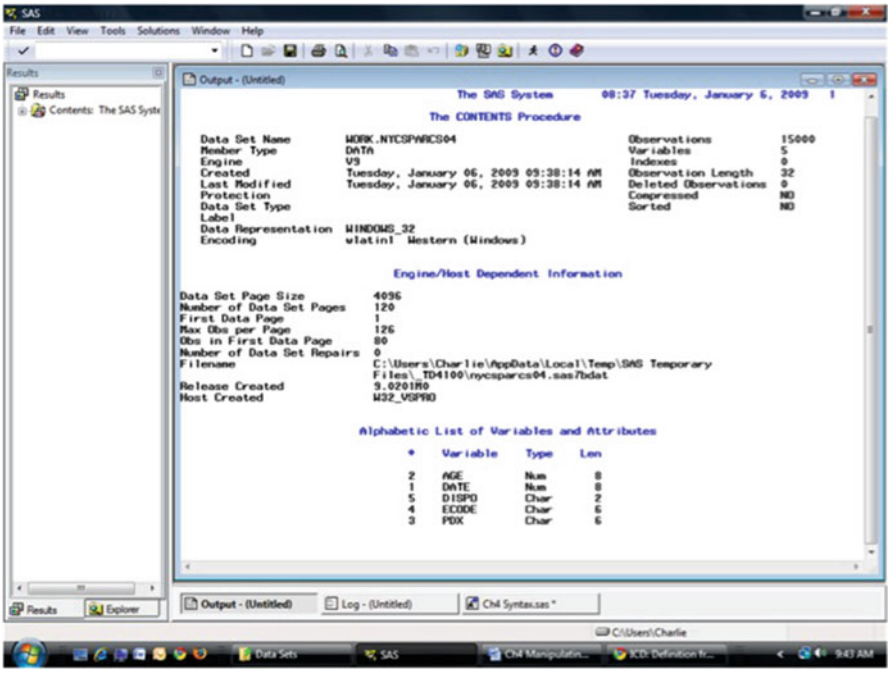


Fig. 5.5 PROC CONTENTS for data file created with IF-THEN-ELSE variables

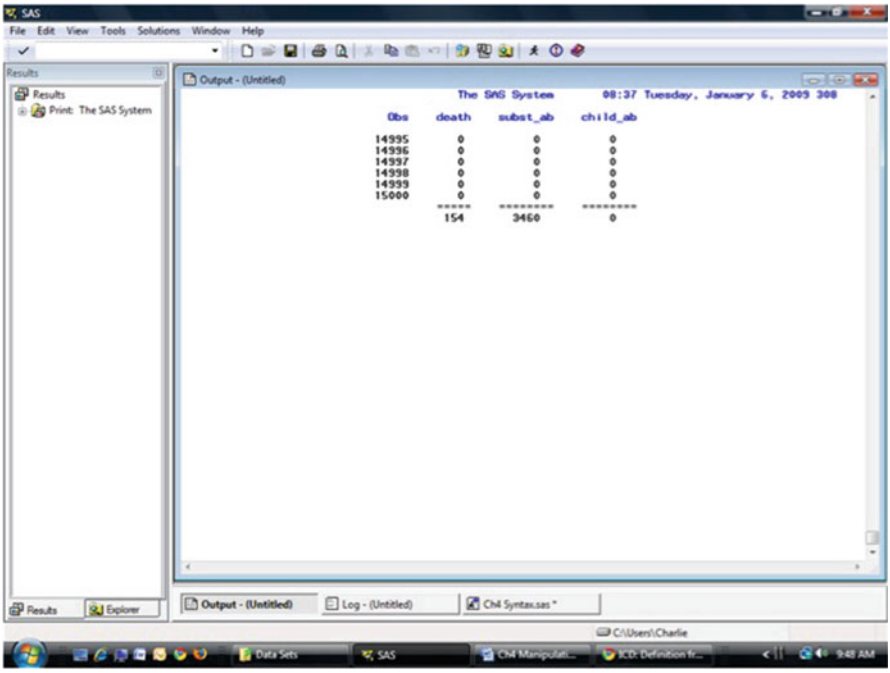


Fig. 5.6 PROC PRINT for data file created with IF-THEN-ELSE variables

5.6 Conditional Expressions Using a Restricting IF Statement

An IF statement alone, as part of a DATA step, will restrict data to a subset by allowing only observations that meet the criteria. This could be useful if you have a large, unwieldy data set, and you are only interested in part of it. Say, for example, you are interested in outcomes in children but have a data set for all ages. You could create a new data set with this syntax, which creates a new data set called "kids" based on an existing data set called "allAges" by using a restricting IF statement based on the variable "age":

```
DATA kids;
SET all_ages;
IF age < 18;
RUN;
```

As in most things SAS-related, there is more than one way to accomplish this task. We could, alternatively, have used a DELETE statement to do the same thing:

```
IF age > = 18 THEN DELETE;
```

5.6.1 Restricting Variables Read into a New Data Set

The above techniques will restrict the observations read into a data set. You may, though, want to drop or keep certain variables read into a new data set. For this purpose, SAS has the very intuitively named

```
DROP=( )    KEEP=( )
```

which can be used as part of a DATA step.

The following syntax creates a new data set named "new" based on an existing data set named "old." We tell SAS not to include two variables (var1 and var2) and to create a new variable named "total" based on the sum of var3 and var4:

```
DATA new (DROP=var1 var2);
SET old;
total=var3 + var4;
RUN;
```

We could, alternatively, have used ($KEEP = var3\ var4$) as part of the DATA statement to accomplish the same thing. It all depends on how many variables you are dropping and how many you are keeping.

5.7 Conditional Expressions with SAS Dates

We've encountered SAS dates before, but they deserve a bit more attention. Recall that SAS dates are essentially numeric variables indexed to January 1, 1960. As we have seen previously, they can be formatted in any number of ways, but they are at their core simply a number. One nice thing about this for epidemiologists is that we have an easy way at getting at and manipulating a time variable.

By putting quotes around a date and following it with the letter d, you are telling SAS to read and turn that date into the corresponding numeric value. So, for example,

```
4dec2000d.
```

is considered by SAS to be the number 14958.

In this way, you can do things like string dates to create ranges like

```
14dec2000d-19dec200d.
```

and use them as part of conditional expressions. The following example demonstrates how useful this can be.

5.7.1 Using Dates to Subset the 9/11 Data

Say, we're interested in whether the terrorist attacks of September 11, 2001, in New York City were so stressful as to put individuals at increased risk of such outcomes

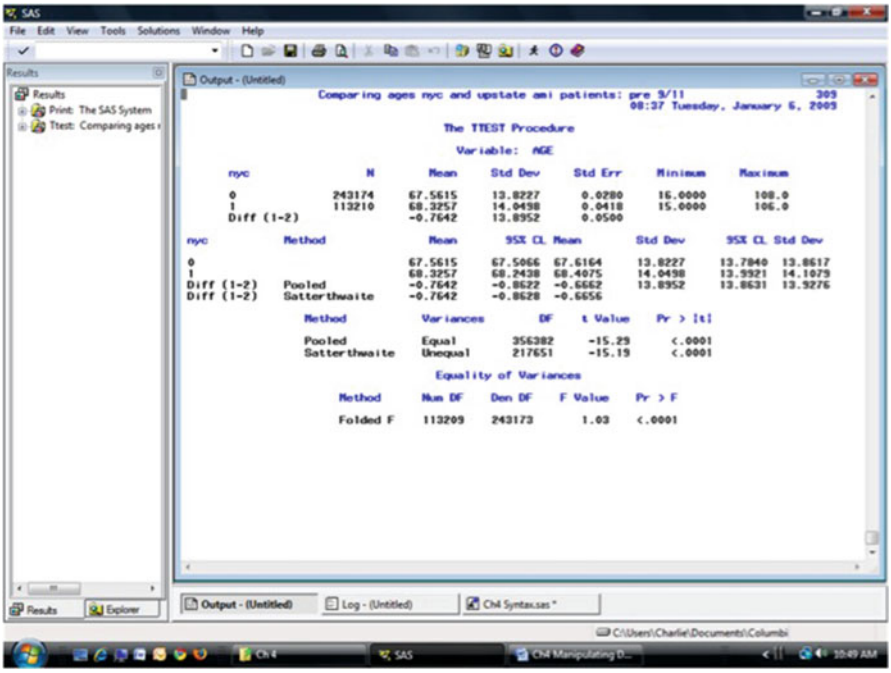


Fig. 5.7 PROC TTEST results (before terrorist attacks)

as acute myocardial infarctions (heart attacks). We might, as one step, see whether compared to non-New York City residents, younger folks in New York City were more likely to suffer AMI following the terrorist attacks of September 11 than they were before the terrorist attacks. We can use the data from the file ch5demo4 to help answer that question. This is a file we encountered before.[8]

The following syntax requests a t-test on the continuous variable age, comparing New York City residents to non-New York City residents for the time period before September 2001. Don't worry about the t-test syntax. Note, though, the conditional expression restricting observations to just those that occurred prior to October 2001:

```
proc ttest data=ch5.ch4demo4;
class nyc;
var age;
where date < '1oct2001'd;
title 'Comparing ages nyc and upstate ami patients: pre 9/11 ';
run;
```

Our results page looks like this (Fig. 5.7).

[8]The data set of all myocardial infarction admissions in New York City from 1994 to 2004 we used in Chap. 1.

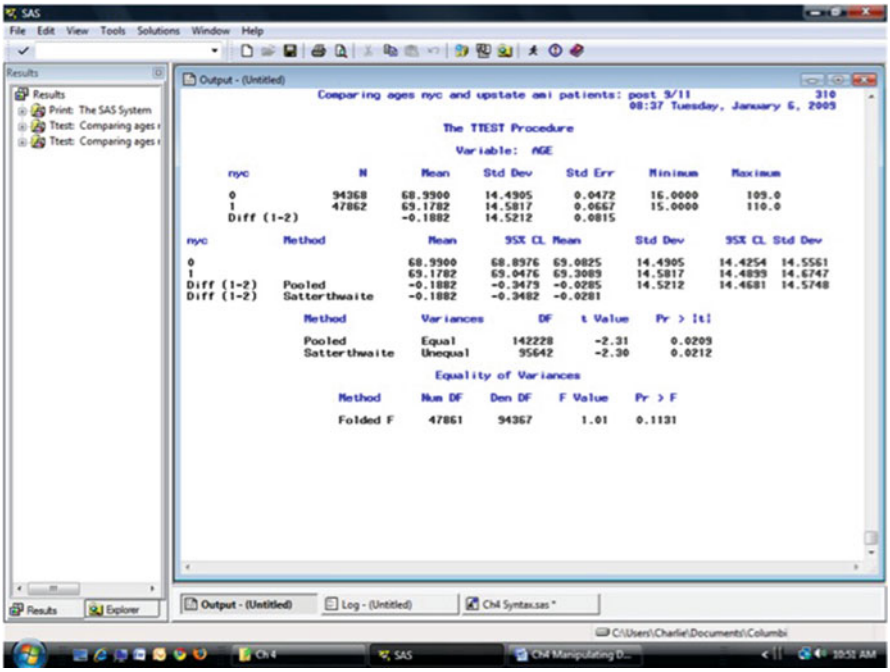


Fig. 5.8 PROC TTEST results (after terrorist attacks)

The mean age of AMI patients in New York City in the period preceding the terrorist attacks was 68.3, compared to the mean age of 67.5 for non-New York City residents. Let's run syntax for the postattack period:

```
proc ttest data=ch5.ch4demo4;
class nyc;
var age;
where date > '1oct2001'd;
title 'Comparing ages nyc and upstate ami patients: post 9/11';
run;
```

There is very little difference, leading us to believe that the terrorist attacks had little effect on hospital discharges for AMI for city vs. non-city residents (Fig. 5.8).

WHAT TIME IS IT?

Believe it or not, you may someday need to find out what the internal SAS value is for a date. Here's how to do it:

```
data _null_; *tell SAS not to create a data set;
date='1jan2001'd; *the date you want;
put date=;
run; /* will show up in log */
```

Problems

5.1. Concatenating Data Sets

Return to the chapter section where we added the file of New York City to non-New York City deaths. Create the totDeaths file in your work library as described in the example. Using the SAS Explorer window, locate the totDeaths file.

- What is striking about the data set?
- What do you notice about the variable for total deaths?
- Correct this problem by using a SET statement to create a total death variable in a new data set also called totDeaths. Did this solve the problem?
- Look at the log after creating the new data set. What warning did you receive?

5.2. Merging and Performing Operations on Data Sets

Using data sets Ch5demo1, Ch5demo2, and Ch5demo3, perform the following operations and answer these questions:

1. Add data from Ch5demo1 to the data from Ch5dem2 to create a complete data set called MERGE1.
   - How many observations were read in from Ch5demo1?
   - How many from Ch5demo2?
   - How many observations and variables are in the new data set? (Hint: Look at your log file.) Does this appear correct? How could you determine if an error occurred?
2. Merge the data from Ch5demo3 with the data from the newly created MERGE1 data set. Name the new data set MERGE2. You will need to decide by which variable you should merge.
   - What do you have to do with the two data sets before merging them?
3. Working with the file MERGE2, calculate the death rate per 100,000 population in each zip code. Name the new file MERGE2CALCS.
   - What is the death rate for zip code 11566?
   - What was the death rate for zip code 10032? (Hint: Use the SAS Explorer window and look in your Work Library.)
   - What conclusion do you draw from this comparison? (Hint: Zip code 10032 is in Northern Manhattan approximately 10 miles from ground zero; ZIP code 11566 is on Long Island about 40 miles from ground zero.)
   - Assuming that most if not all the victims of 9/11 were of working age (say, between the ages of 20 and 64), create a variable that lists the death rate per zip code for working age individuals. What are the new rates for workers for zip codes 11566 and 10032? What conclusions can you draw from this new information?

Part II
Descriptive and Categorical Analysis

The first step in any epidemiological analysis should be describing your data. If you're working with existing data, you will uncover anomalies, errors, and idiosyncrasies that must be addressed early in your analysis. It is only after cleaning your data that you should turn to looking at potential patterns and associations. Much of epidemiology involves categorical outcomes like morbidity and mortality, so we turn to this task first. If, though, your main outcome of interest is continuous, you will want look at the material in Part III of this text.

Chapter 6
Descriptive Statistics

Abstract In this chapter we introduce those procedures that are most likely to be of benefit at the earliest stages of epidemiologic analysis. Some relatively simple procedures (MEANS, FREQ, TABULATE) can be used to get descriptive information about your data. Sometimes this may be all you are interested in. More likely, you will use this information to inform and guide further analyses. We will defer for the moment some of the theory underlying these analyses and will return to the topics of categorical and continuous analyses in future chapters.

6.1 PROC MEANS

The humble PROC MEANS returns descriptive statistics such as the means and standard deviations of continuous variables. The main options for PROC MEANS are VAR (to select variables) and CLASS (to group observations). MAXDEC included as part of the data statement is a useful option to limit the number of decimal places. Syntax for PROC MEANS looks like this:

```
PROC MEANS data=your.data MAXDEC=2;
VAR age height weight;
CLASS gender;
```

The variables specified with VAR are all continuous and numeric, and we have used the CLASS option to request that the results be grouped by gender.

The data file ch5demo1 is a sample of 1,000 of the California myocardial infarction observations we looked at earlier. The following syntax runs MEANS on the age variable grouped by gender:

```
proc means data=ch6.ch5demo1;
var ageyrs;
class sex;
run;
```

C. DiMaggio, *SAS for Epidemiologists: Applications and Methods*,
DOI 10.1007/978-1-4614-4854-9_6,

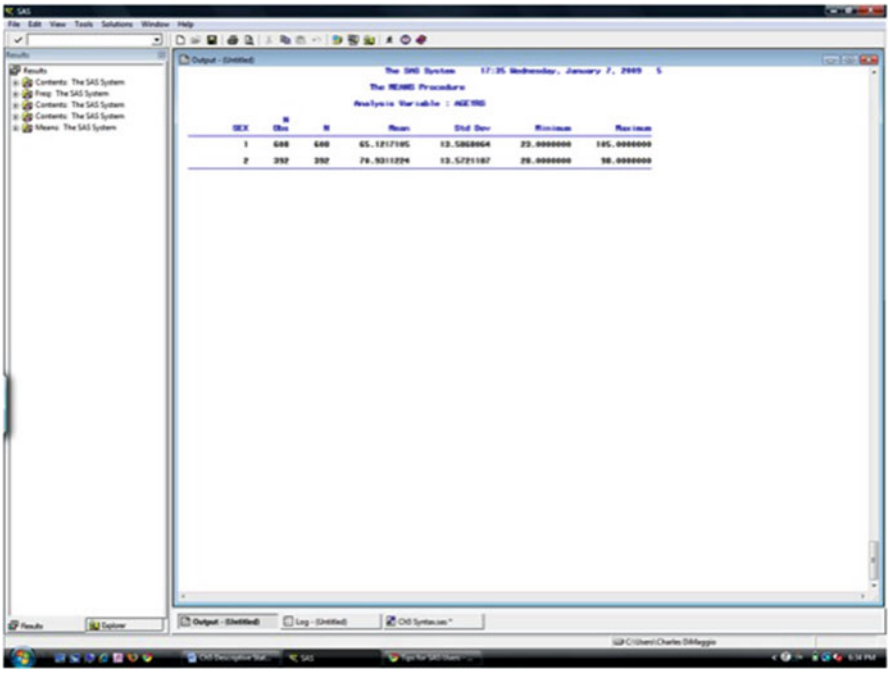


Fig. 6.1 PROC MEANS results

In the results window, we are given the count, mean age, standard deviation, as well as minimum and maximum values for age for each gender. This is useful information in and of itself, but in early stages of data analysis, it is most helpful for identifying problematic observations. For example, what are we to make of the 105 year old? We may want to be sure this was coded correctly (Fig. 6.1).

6.2 PROC FREQ

The FREQ procedure returns frequency counts and cross tabulations of *categorical* variables, i.e., those variables that name or categorize things, like gender, diagnostic category, or exposure status. All those two-by-two tables we learn about in our introductory epidemiology courses can be most readily analyzed with PROC FREQ.[1]

Rather than a VAR statement to identify the variables we want to analyze, in PROC FREQ we use a TABLES statement to identify the categorical variable(s) to analyze. As part of the TABLES statement, you can set cross tabulations. Use an

[1] We will go into more detail in upcoming chapters.

asterisk (*) to connect the two variable you want to cross tabulate. A key PROC FREQ option (for an epidemiologist) is the MEASURES command which, for a 2×2 table, will return odds ratios. Another useful option is NOCUM, which suppresses some of the voluminous cumulative frequencies and percentages that can complicate your output.

Here is an example of PROC FREQ syntax requesting simple (separate) frequency tables for the two variables exposure and mortality:

```
PROC FREQ data=your.data;
TABLES exposure mortality
RUN;
```

An example of PROC FREQ syntax looking at the association between some exposure and an outcome like mortality would be

```
PROC FREQ data=your.data;
TABLES exposure*mortality / MEASURES;
RUN;
```

The only difference between this and the previous syntax is the asterisk.

Let's return to our ch6demo1 file and calculate an odds ratio for the association of gender with race among these myocardial infarction patients:

```
proc freq data=ch6.ch6demo1;
tables sex*race2 / measures;
run;
```

The first page of results (not pictured) will give us a 2×2 table and some statistics. The final page of results returns an odds ratio of 0.9 with a 95% confidence interval 0.7–1.2. We will spend a considerable amount of time with PROC FREQ when we consider the analysis of categorical data later.

6.3 PROC TABULATE

The TABULATE procedure is a sort of hybrid of MEANS and FREQ. It's useful in the analysis of continuous variables grouped by some classifying categorical variable. It is, actually, a very powerful procedure, and I've been told you could almost sustain a career based on an intimate knowledge of PROC TABULATE. For our purposes, a cursory understanding is sufficient to give us some tools that are useful for epidemiological investigation.

The general form for proc tabulate is:

```
PROC TABULATE data = dataset;
        VAR continuous_variable
        CLASS categorizing_variable
        TABLES  page, row, column
RUN:
```

As in PROC MEANS, a VAR statement is used to indicate a continuous variable you want to analyze, and CLASS is used to indicate a categorical or classifying

variable by which you want the continuous variable grouped. A PROC TABULATE statement must have at least a CLASS or a VAR statement, though not necessarily both.

The real power of TABULATE is in the TABLE statement. Whereas in PROC FREQ the TABLE statement is fairly simple with dimensions indicated by an asterisk (*), a TABLE statement in PROC TABULATE can set up a multidimensional table of columns, rows, and pages. The TABLE statement in PROC TABULATE has its own set of unique operators.

A *comma* indicates a new table dimension in the order of *page-row-column*.[2] So, if only one variable is specified, it will be arranged in columns. If two variables are specified and they are separated by a comma, the right-most variable will be in columns arranged by the left one which will determine the rows. If three variables are specified each separated by commas, there will be a page for each value of the left-most variable, values of the middle variable determines the rows, and the right-most variable will be in the columns.

The following code will give you a table with a column of counts for each type of diagnostic-related group for our ch5demo1 file (Fig. 6.2):

```
proc tabulate data=ch6.ch6demo1;
          class drg;
          table drg;
run;
```

There are two additional operators for the TABLE statement. A *blank space* is a command to concatenate table information. An *asterisk* is a command to cross-nest, or subgroup information. This may be best understood through some examples.

Here, we use a blank and the key word "all" to get a total (Fig. 6.3):

```
proc tabulate data=ch6.ch6demo1;
          class drg;
          table drg all;
run;
```

Next, to get a two-dimensional table of two categorical variables (here, diagnostic-related group and race), we first specify the categorical variables with a CLASS statement, then use a comma as part of the TABLE statement to set up the dimensions of the table. Notice that we are also requesting a total count by including a space and the word "all" (Fig. 6.4):[3]

```
proc tabulate data=ch6.ch6demo1;
          class drg race2;
          table drg all , race2 all;
run;
```

[2]Some users of PROC TABULATE believe it is easier to read the comma-delineated TABLE specifications (page-row-column) from right to left.

[3]Note that a simple PROC FREQ with TABLES drg*race2 would basically achieve the same result. Although this is an example of the frequent situation where there is more than one way to achieve something in SAS, you will find, though, that if your interest is in creating tables, PROC TABULATE affords you much more flexibility.

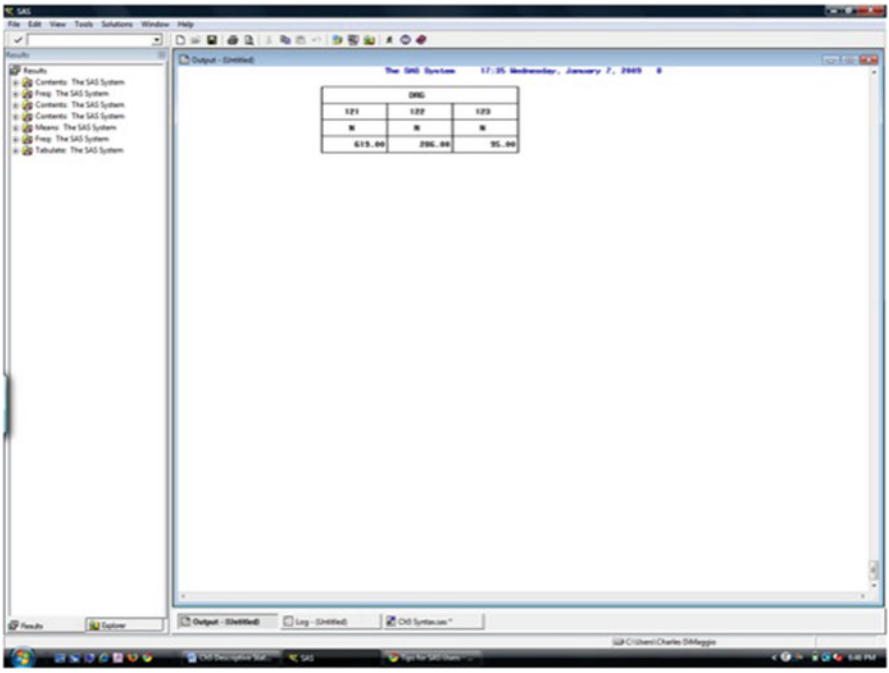


Fig. 6.2 Simple PROC TABULATE results

Up to now, we could pretty easily have created these tables with a simple FREQ statement. We have not yet included a VAR statement to identify a continuous variable upon which to conduct analyses. Let's do so now. The default analysis for a continuous variable is a *sum*, but a number of other analyses are available as part of PROC TABULATE. Here, we request analysis of an age variable in a table of diagnostic-related groups by race (Fig. 6.5):

```
proc tabulate data=ch6.ch5demo1;
         class drg race2;
 var ageyrs;
         table drg, race2*ageyrs;
run;
```

Of course, the sum of ages is not very helpful. This is just a first, simple example. We can, though, request a number of statistics by following the writing a TABLE statement with the continuous analysis variable (which may be nested in a categorical variable) with an asterisk and the desired statistic (Fig. 6.6):

```
proc tabulate data=ch6.ch6demo1;
         class drg race2;
 var ageyrs;
         table drg, race2*ageyrs*mean;
run;
```

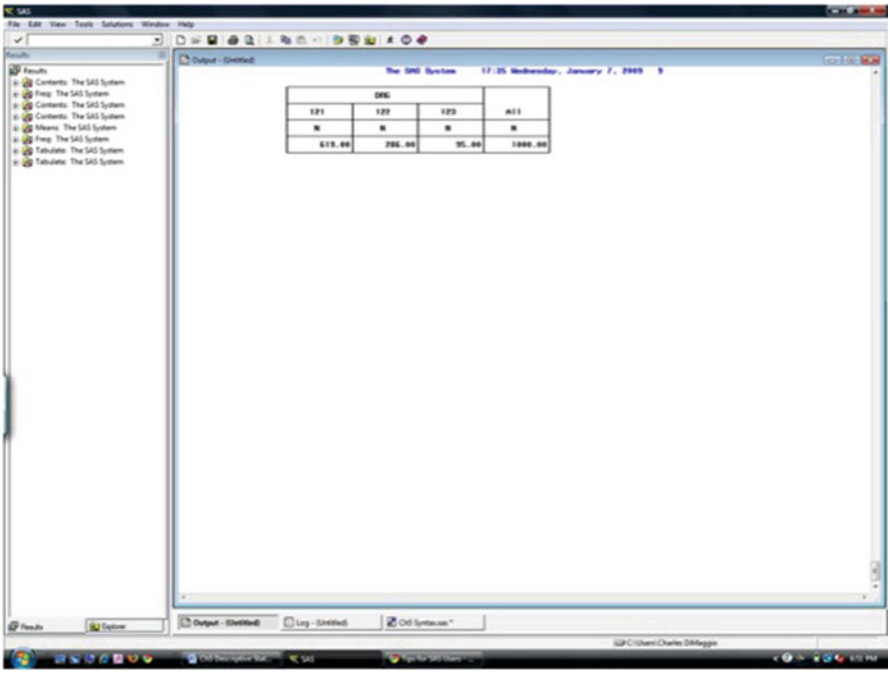


Fig. 6.3 PROC TABULATE with a total

Other useful statistics you can ask for are *median, min, max, std* (standard deviation), and *nmiss* (number of missing observations).

6.3.1 Using TABULATE for Surveillance Data

One epidemiologically useful application of PROC TABULATE is create data sets based on tables that sum outcomes or diagnoses over some time period for surveillance. The following syntax will accomplish that task:

```
/*proc tabulate to create output data set for surveillance*/
proc tabulate data=ind.sparcs94; /*individual-level data*/
where nyc=1; /* subset to geographic area */
class month; /* specify surveillance period using
 variable previously created with MONTH() function */
var injury ami child_ab; /* specify diagnoses to monitor
                          (have to be numeric) */
table month, injury ami child_ab; /* table specifications
                            (default is sum or total) */
ods output table=sparcs94; /*create output dataset from table*/
data sparcs94; /*clean up dataset*/
    set sparcs94;
drop _type_ _page_ _table_; run;
```

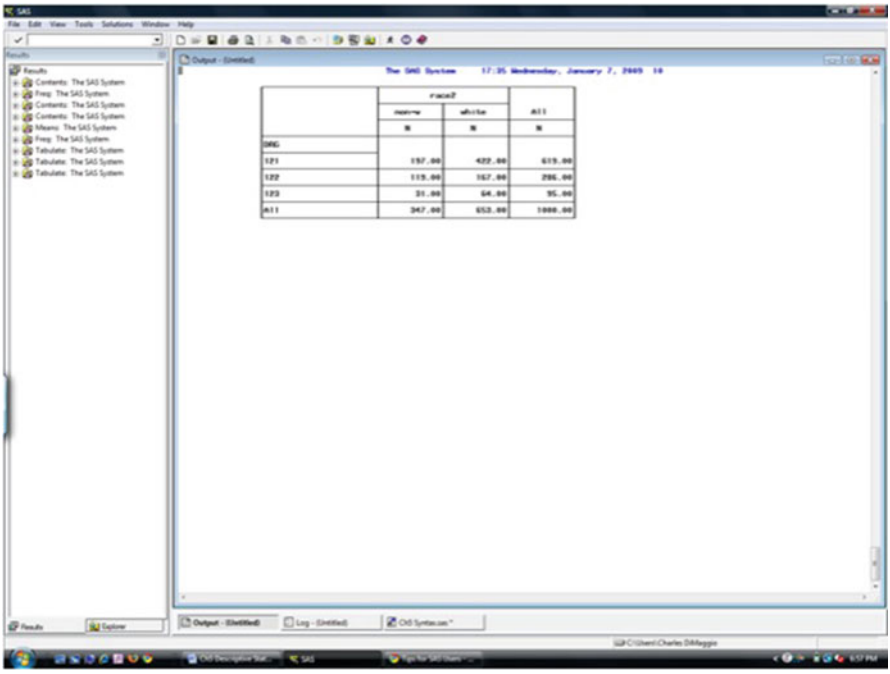


Fig. 6.4 PROC TABULATE with an additional dimension

A VICIOUS CYCLE

If all this TABULATE syntax seems terribly complicated and different from the rest of SAS syntax, well, it is. In fact, PROC TABULATE grew out of a separate programming language that was adopted whole cloth by the makers of SAS. They didn't tinker too very much with it, because TABULATE is quite powerful in its own right. I invariably have to come back to the documentation for PROC TABULATE just to remind myself how to use it. But that might be because I don't use it often enough. And perhaps because I don't use it often, I have to look up the syntax. It's a vicious cycle. But there are times when nothing but PROC TABULATE will do. For example, if you want a straightforward way of summarizing across time periods for surveillance. So it's good to keep it as an arrow in your quiver.

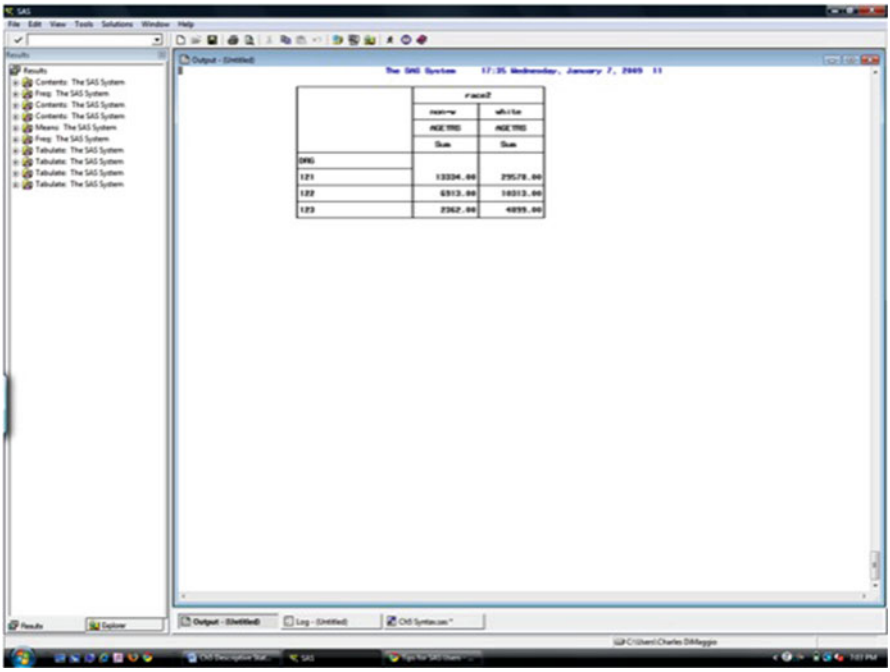


Fig. 6.5 PROC TABULATE with a continuous variable

Problems

6.1. Reading Data into SAS

In this series of exercises you will work with New York City hospital discharge data to walk through the steps involved in the initial analysis of health data. We begin by reading the data into SAS.

The data is located on the NYCSPARCS05.txt file. You will first need to read it into SAS with a DATA-INFILE-INPUT statement. Create a temporary file in your work directory. You can name it anything you like. Specify a long record length (LRCL=450) and have SAS apply default missing values (MISSOVER). Read in all the observations, but limit the variables to the date of admission, patients age, county and zip code of residence, sex, race, primary diagnosis, and length of stay. You can find the location and format of these values in your SPARCS documentation and layout file. To save time, I've looked them up for you:

```
Variable     Location  Informat
DATE         @18 yymmn6.
AGE        @44  3.
COUNTY  @50  $CHAR2.
ZIP             @52  $CHAR5.
```

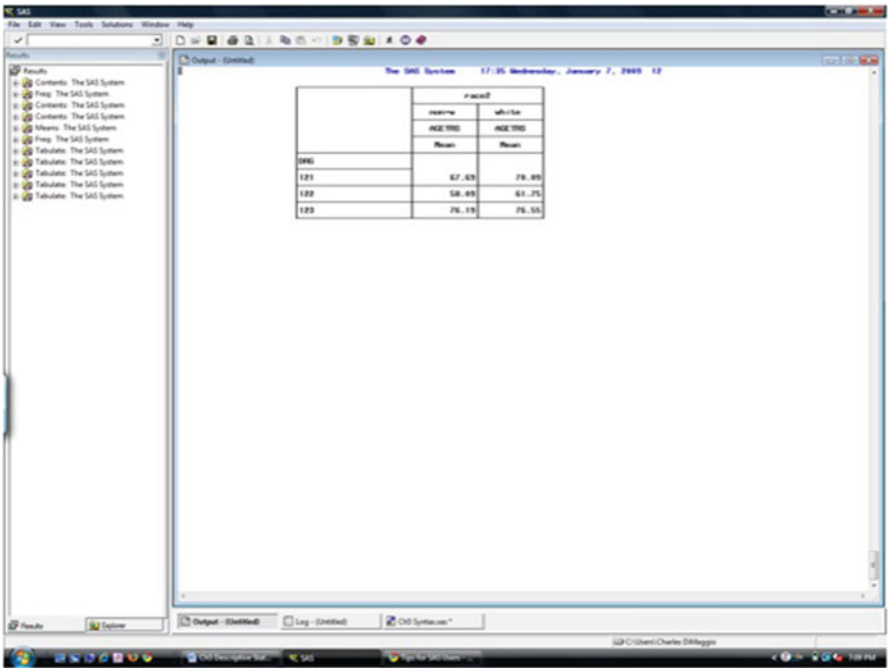


Fig. 6.6 PROC TABULATE with summary statistic for a continuous variable

```
SEX          @57   $CHAR1.
RACE         @58   $CHAR2.
PDX          @71   $CHAR6.
LOS        @349   4.
```

6.2. Create New Variables from Existing Variables

Now that you have a SAS data set (did you check your log and print out a few lines of data?), you can create some new variables and change existing ones so they are more amenable to analysis. Time is a crucial variable in epidemiologic studies. We'd like to have some sense of when, during the year, events occur. Create a month variable from the existing date variable. There is a very nice SAS function that will do this for you. Try to find it online or in your documentation.

For analytic purposes, it is often useful to have a numeric indicator variable for categorical attributes. Create a numeric variable called GENDER based on the SPARCS character variable SEX. There are a number of ways to do this, but in this instance a series of simple IF-THEN-ELSE statements to create two new variables (male and female) will suffice.

Here's the code to do it:

```
IF sex='M' THEN male=1;
else male=0;
IF sex='F' THEN female=1;
ELSE female=0;
```

Use this same approach to create numeric 0/1 indicator variables from the SPARCS character RACE variable for white (01), black (02), asian (04), other_race (88), and unknown_race (99). Again, these codes area available in the SPARCS documentation.

Founding father of epidemiology William Farr is reputed to have said death is a fact. All else is inference. Create a 0/1 mortality indicator variable from the existing DISPO (disposition) variable. Find the appropriate disposition code for death in your SPARCS documentation.

The outcome variable in this data set is, essentially, the primary discharge diagnosis, PDX. This is in ICD9 format, which is a series of 3–5 numbers. If your research interest is extremely specific, there may be just one or two codes in which you are interested. Sometimes, someone else has done similar work and you can find the code online. CDCs National Center for Health Statistics is often very good that way. More likely, you will need to comb through the ICD9 documentation to identify all codes that relate to the outcome. If you have no clinical experience in the area (or at all), you would do well to get input from clinical colleagues on your list of codes. In this example, we are interested in substance abuse. Use the list of ICD9 codes in the document "Substance Abuse ICD" with an IN lookup table as part of an IF-THEN-ELSE statement to create a 0/1 indicator variable for substance abuse.

6.3. Using PROC MEANS

What is the oldest age in this group?

6.4. Using PROC FREQ

Run PROC FREQ on county, race, and sex. What, if anything, is unusual or striking about the county results? Which county represents Manhattan?[4] You will have to look in your documentation to answer this question. Where does the largest percentage of non-NYC residents come from? (Again, use your documentation) What, if anything, about the race results may cause some concern about the reliability and validity of that variable?

6.5. Using PROC TABULATE

Use PROC TABULATE to look at substance abuse.

Create a summary table of the number of substance abuse discharges each month and the percent of the total monthly discharges they represent. This approach can be used to sum across monthly or weekly data to perform surveillance activities in health departments.

[4]New York City consists of five counties (Manhattan, Bronx, Kings, Queens, Richmond).

Limit your observations to just those in New York City counties with a WHERE statement (where county in (‘58’ ‘59’ ‘60’ ‘61’ ‘62’);). Your CLASS variable will be the month variable you created. The VAR variable is the substance abuse variable you created. Your TABLE statement should request two dimensions with the left-most dimension (row) your month variable. Since you want both the total number of substance abuse diagnoses each month, as well as the percentage of total diagnoses they represent, you will need two items on the right side of the comma: subst_ab*sum subst_ab*pctsum. Note that they are separated by a space.

Create a table to determine the total number of substance abuse diagnoses for each county in New York City as well as the percent of the total represented by that number. Here’s some code to help you:

```
proc tabulate data=nycsparcs; /*create dataset*/
where county in ('58' '59' '60' '61' '62');
class county;
var subst_ab;
table county, subst_ab*sum subst_ab*pctsum;
run;
```

Assuming all counties receive the same amount of resources from the city for substance abuse (not so in reality), based on this analysis, which county would appear to require more resources for substance abuse? Which might require less? What other variables might affect these results? What other information might you want to determine?

Finally, include the following statement to create a table from your output, which can be used for graphing:

```
ods output table=subst;
```

Chapter 7
Histograms and Plots

Abstract SAS has some very powerful graphing procedures that are well worth exploring and utilizing. In this chapter we introduce some of that capability by presenting syntax for histograms and plots.

7.1 Introduction

In the interests of full disclosure, I must admit I seldom use SAS to plot or graph data. I generally use R for its impressive and (to my mind at least) intuitive graphic features, or export the data into Excel and Microsoft's point and click approach. I think, though, this is less a reflection on SASs graphic capabilities, which are by many accounts protean, than it is on my decreasing willingness to learn new things when old ones seem to suit my need.

SAS does have some very neat graphing features, and they are well worth exploring and utilizing. And, more often than not, it is more convenient to create charts and graphs in the same program in which you are conducting analyses. In this chapter I present the merest hint of what SAS is capable of, specifically how to produce histograms and plots.

7.2 PROC GCHART for Histograms

You can easily recognize SAS graphing procedures because they have a G in front of them, for example, PROC GCHART and PROC GPLOT. PROC GCHART is the SAS tool to produce histograms. For categorical variables, you'll get a bar for every value of that variable. For continuous variables, SAS will determine the values.[1]

[1] Although it is never a good idea to let a machine dictate something this important.

C. DiMaggio, *SAS for Epidemiologists: Applications and Methods*,
DOI 10.1007/978-1-4614-4854-9_7,

In its simplest form, you invoke PROC GCHART, identify the data set, and then specify whether you want vertical bars (VBAR) or horizontal bars (HBAR) for the listed variables. You can request a pie chart by specifying PIE instead of VBAR or HBAR.

SAS produces charts in a separate graph window, with a name like

```
GRAPH1.GSEG.CHART1.
```

With that window active, you can save the image as a jpeg, a gif, etc. In many systems, you can just right click on the figure and choose

```
Edit --> Copy --> Paste
```

I find saving the file as an .emf (enhanced metafile) works best for inserting into windows documents.

It is not an error when you see PROC GCHART still running at the top of the editor screen even after you've created your chart. It just means that the procedure is active. Running other procs will exit you from GCHART, or can type "quit."

You can use GCHART to perform some simple analysis of variables by including a forward slash (/) and options on the line of syntax following the variable . For example, "SUMVAR=" identifies a variable to use for a sum or mean calculation for each value of your variable, and TYPE= specifies that the height or length of the bar or size of the slice represents a mean or sum of the analysis variable values.

If requesting a pie chart, some useful options include "FILL=x" to request a cross-hatch pattern, or "EXPLODE=" to request that a certain value of the variable be a slice pulled away from the pie.

A particularly useful option from the perspective of epidemiology is "GROUPVAR=", which allows you to specify that variables be analyzed by some grouping or categorizing variable. This could be exposed vs. unexposed, treated vs. not treated, and cases vs. controls.

As a demonstration, we will work with a file of children, some of whom have a disease of interest (cases) and some of whom do not (controls).[2] As a first pass, say we want to know how many children in the entire sample are male and how many female (Fig. 7.1):

```
PROC GCHART DATA = ch7.ch7demo1;
     VBAR sex;
RUN;
```

Say, though, we are interested in seeing the difference in genders between cases of disease and noncases? A GROUPVAR option will allow us to do that (Fig. 7.2):

```
PROC GCHART DATA = ch7.ch7demo1;
     VBAR sex / groupvar=case_control;
RUN;
```

[2]This is data from an actual study which, for confidentiality purposes, we will not further describe.

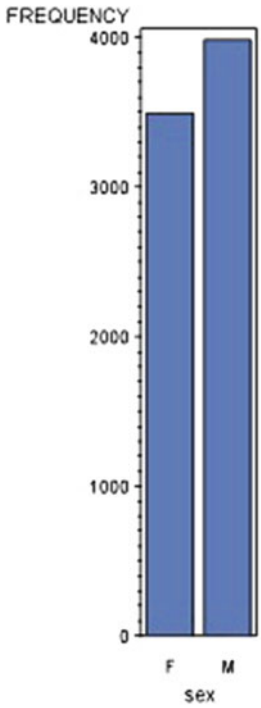


Fig. 7.1 Bar graph with GCHART

By this point you will have realized that the default in GCHART is for all bars to have the same color. You can override this by typing in the somewhat inscrutable command "PATTERNID=MIDPOINT" in the HBAR or VBAR statement (Fig. 7.3).

```
PROC GCHART DATA = ch7.ch7demo1;
     VBAR sex / groupvar=case_control patternid=midpoint;
RUN;
```

There are, of course, many, many other graphing options. I leave it to you to explore them by searching under the topic GOPTIONS.

7.3 PROC GPLOT to Plot Continuous Data

PROC GPLOT is the tool to plot continuous data. Essentially you will be plotting one variable against another to create a scatter plot or perhaps a time series of some kind. The syntax, in its simplest form, is very straightforward:

```
PROC GPLOT DATA=
       PLOT var1 * var2
RUN;
```

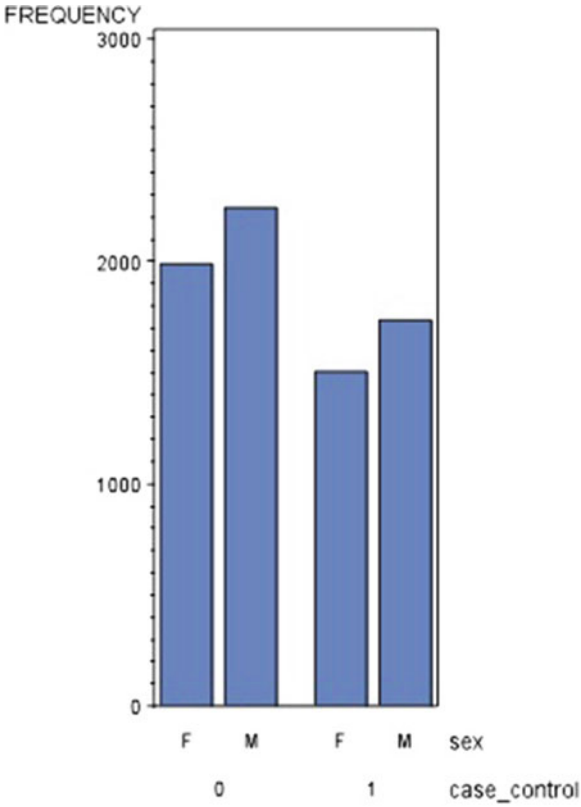


Fig. 7.2 Grouping the bar chart

You can subset the data using a WHERE statement and control symbols, how lines are joined and colors, using a SYMBOL statement. As part of your SYMBOL statements you can include a VALUE= option to choose a particular type of symbol (a plus sign is the default) such as a STAR, DIAMOND, SQUARE, or TRIANGLE. An I= option allows you to choose JOIN (join points with a line) or SPLINE (smooth line) (the I stands for interpolation). You can adjust color and width with C= and W= symbol options.

Note that SYMBOL statements are global (they last for the entire SAS session) and additive (you can append onto previous statements). You will have to use a RESET statement to get back to the default settings. To cancel all your previous SYMBOL statements, type and submit

```
goptions reset=symbol;
```

As an example of how useful PROC GPLOT can be, let's return to our example of acute myocardial infarctions before and following the terrorist attacks of September 11, 2001, in New York City. We learned in Chap. 5 that we can use PROC TABULATE to sum the number of diagnoses over each month-year time period and then use ODS to create an output data set of these summed monthly counts. We now use this data with PROC GPLOT to create a time series plot (Fig. 7.4).

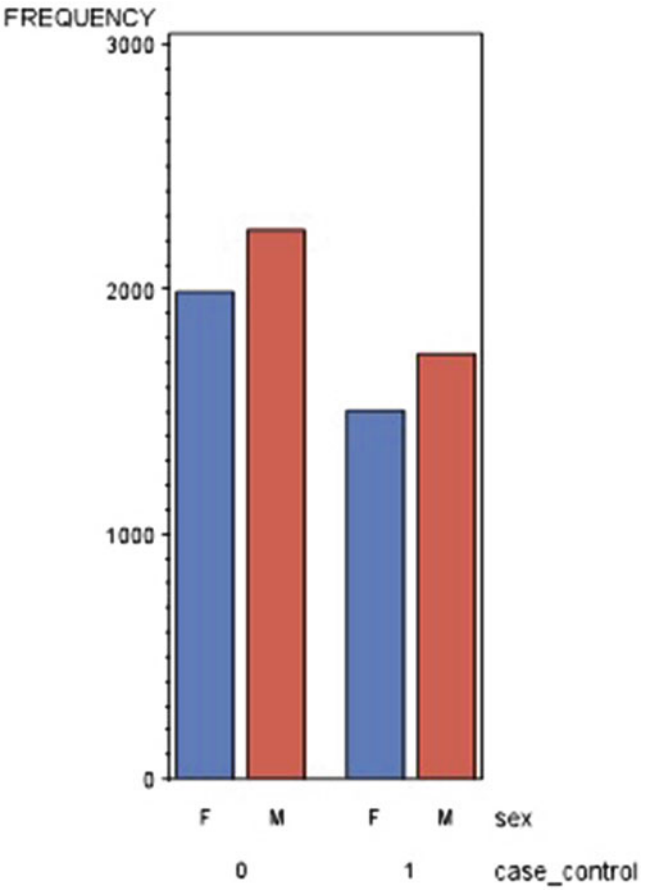


Fig. 7.3 Applying unique colors to the grouped variables

Don't be frightened by the following syntax. The actual GPLOT syntax is actually rather short. Everything else is, believe it or not, bells and whistles associated with fonts, colors, titles, etc. I suggest you do what all the best programmers do at some point in their careers and simply take the code and use it for your own:

```
/*************   AMI Time Series  ************************/

 goptions reset=all;
 goptions cback=white device=win;

 axis1  major = (h=2.0  c=black)
        minor = (h=1.0 c=black)
        order = 0 to 12000 by 1000
```

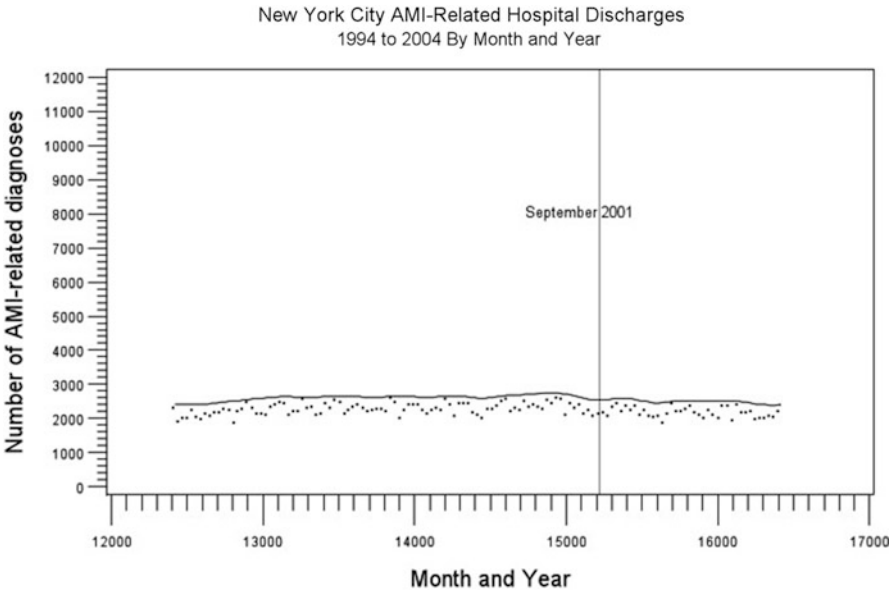


Fig. 7.4 Time series graph for surveillance

```
        label = (h=1.5 a=90 f=arial c=black 'Number of
                  injury-related diagnoses');

axis2   major = (h=2.0  c=black)
        minor = (h=1.0 c=black)
        /*order = 1 to 132 by 6*/
        label = (h=1.5 f=arial c=black 'Month and Year');

proc gplot data= ch7.ch7demo2;
plot ami\_sum * date / href=15219
                       vaxis = axis1
                       haxis = axis2;
     symbol v='.' f=arial h=2 i=sm30 c=black;
     note h=1.0 move=(59,60)pct 'September 2001'; * percent
       (over, up);
     title h=3.0 c=black f=arial j=center
           'New York City AMI-Related Hospital Discharges';
     title2 h=2.0 c=black f=arial j=center
           '1994 to 2004 By Month and Year';
run;
quit;
```

IT KEEPS RUNNING

GCHART and GPLOT are procedures that will keep running in the background until you type the command "quit."

Even then, SAS keeps on appending graphs in the graph window. The only way to get rid of old graphs is to open the "G seg" folder in the Explorer Window (found in the work library) and delete the graphs you don't want. You can also delete the entire Gseg folder to get rid of all the graphs. SAS will recreate it the next time you need it.

Problems

7.1. PROC GCHART

Return to or recreate the SPARCS data set we worked with in the exercises to Chap. 5. Write and run a SAS program to create both vertical and horizontal bar charts for the variable "race." Is there any apparent advantage to one graphic (vertical vs. horizontal) versus the other?

Add an option to your syntax for the race histogram to include the mean length of stay for each race. (Hint: you will need to include a SUMVAR= and a TYPE= option on the VBAR and HBAR statements.) Create a pie chart of race, including the mean length of stay by race. Emphasize race category 02. (Hint: you will need a FILL= and an EXPLODE= statement.)

7.2. PROC GPLOT

For this exercise, imagine you work for the New York City Department of Health and Mental Hygiene. You are asked to conduct substance abuse surveillance. One approach could be to look at the total number of substance abuse diagnoses in New York City hospitals and plot them by some time period.

We'll use PROC TABULATE and PROC GPLOT to do that. Working again with the SPARCS data set from above, you will first write a PROC TABULATE statement that sums substance abuse observations by month. Limit your observations to just those in New York City with a subsetting WHERE statement (hint: NYC counties consist of 58, 59, 60, 61, and 62). Use a CLASS variable to group observations by "month." SUM the number of substance abuse diagnoses by using the 0/1 "subst_ab" variable you created when reading in the data. Your TABLE statement should have two dimensions: month and subst_ab*sum. Write the syntax to create this table and run it.

Your next step is to create a data set based on the results of this table so you can read it into PROC GPLOT. Fortunately, SAS provides an easy way of doing this

with ODS. The following syntax will create a data set called “subst04” in your work library based on the table you created in PROC TABULATE:

```
ODS output table=subst04;
```

Include it at the end of your PROC TABULATE statement, just before your RUN statement.

Finally, write a PROC GPLOT statement to plot the number of substance abuse diagnoses per month. Define your data as the subst04 data set you created as part of your PROC TABULATE run. For your PLOT statement specify that you want to plot the variable “subst_ab_Sum” (this is a variable automatically created as part of your PROC TABULATE run) by the variable “month.” For your SYMBOL statement, specify a red diamond of width 5 joined by a spline.

What do you notice about the vertical axis for this plot? Is the spline an effective way of presenting the data?

Add the following vertical axis option as part of the plot statement:

```
/ vaxis= 0 to 5000 by 200
```

Rather than a spline, specify a simple join as part of your SYMBOL statement. Change the color to blue and the width of the symbol to 2.

Chapter 8
Categorical Data Analysis I

Abstract In the next two chapters we consider the kinds of categorical outcomes frequently encountered in epidemiological practice. Categorical variables are those that take on discrete values only. When there are only two possible values, such as survival vs. death, exposed vs. unexposed, or diseased vs. non-diseased, we can refer to them as dichotomous. We will encounter them again as potential explanatory or exposure variables when we discuss ANOVA and dummy variables in linear regression. We now, though, consider them exclusively as both exposures and outcomes. When both our exposure and outcome are dichotomous categorical variables we can apply the classic epidemiological 2×2 table.[1]

8.1 Introduction to Categorical Outcomes

Categorical variables differ in many important statistical ways from continuous variables, and our approach to analyzing them must adapt considerably. For example, they are quite clearly not normally distributed, an assumption underlying many of the approaches we will consider when we analyze continuous variables. When variables can take on, say, only two values, scatter plots certainly don't make very much sense.

SAS comes with many rich tools to work with noncontinuous, non-normally distributed outcome data. For our purposes, we will look at some very basic but potentially very informative approaches, such as arranging data in tables and comparing frequencies across rows, looking at cross tabulations of two categorical variables, and stratifying such a cross tabulation by a third or even fourth variable to assess confounding. In terms of SAS, we will be working almost exclusively in PROC FREQ using "crosstabs."

[1] Practice epidemiology long enough and you may be tempted to collapse the entire universe into a 2×2 table. Its an occupational hazard.

C. DiMaggio, *SAS for Epidemiologists: Applications and Methods*,
DOI 10.1007/978-1-4614-4854-9_8,

8.2 Associations

In categorical data analysis, we are very interested in the idea of associations. In a very commonsense way, an association can be defined as one categorical variable changing in value when the level or value of another variable changes.

Say, for example, we are interested in whether mood is associated with weather conditions. We begin by defining our variables. Say we are going to take a very "2×2" approach to this question and define mood as either "happy" or "sad" and weather as either "nice" or "crummy." We can arrange our data in a classic 2×2 table (Fig. 8.1).

With just a brief glance, we can see that the row percentages are the same across the values of our "exposure" variable.

Let's say though, we find the following results (Fig. 8.2).

Again, just looking at the row percentages indicates that our "outcome" variable of mood seems to vary across the row categories of our "exposure" variable of weather and so may be associated with it. We will, of course, want to quantify any such association and account for the possibility of chance affecting our results. Much of categorical statistical analysis, and the remainder of this chapter, involves how we go about doing that.

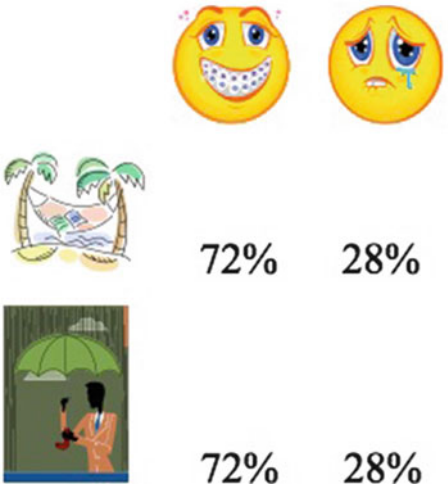


Fig. 8.1 A 2×2 table with no association

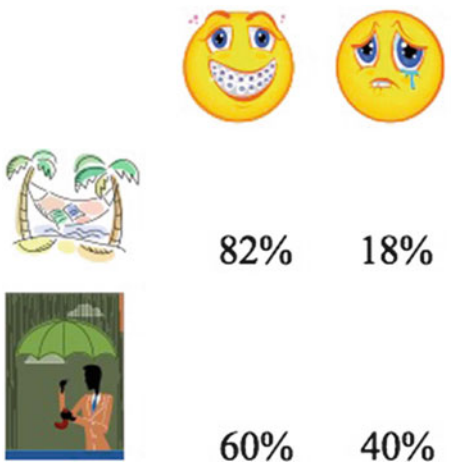


Fig. 8.2 A 2×2 table with an association

8.3 Examining Frequency Tables

Say we are interested in the effect of gender, socioeconomic status (SES), and age on health-care expenditures. We might begin with simple one-way frequency tables of our variables with the following FREQ syntax:[2]

```
proc freq data=expend;
tables ses;
run;
```

Which returns the following table (Fig. 8.3).

We can make a couple of quick observations. First, SES seems fairly equally distributed across the study sample. Second, we see that SAS has its own way of ordering variable values (alphanumerically) and that this default does not make very much sense for our purposes.

It's usually a good idea to get a sense of your categorical data through such one-way tables. You will, though, move quickly to want to cross classify the categorical variables in your data to search for associations. In PROC FREQ this is accomplished by setting up a cross tabulation (Fig. 8.4).

By default, SAS returns four numbers in each cell: the frequency, the percent of the total, the row percentage, and the column percentage. Let's run some crosstabs on our expenditure data. Notice that we are first running PROC FORMAT for our outcome variable which would otherwise be listed simply as 0 or 1. We then use

[2]To recreate these examples, first run the file Ch7 Demo Data.sas. These data actually have nothing to do with health expenditures. They are from SAS training material and I believe refer to shopping patterns, but they are a convenient way to illustrate some of the approaches to categorical data analysis.

Fig. 8.3 Table cells from PROC FREQ

SES	Frequency	Percent	Cumulative Frequency	Cumulative Percent
High	155	36	155	36
Low	132	31	287	67
Medium	144	33	431	100

	column1	column2	...	columnC
row1	cell1,1	cell1,2	...	cell1,C
row2	cell2,1	cell2,2	...	cell2,C
...	...	...	...	...
rowR	cellR,1	cellR,2	...	cellR,C

Fig. 8.4 Cross classification in a PROC FREQ table

PROC FREQ to request one-way tables for all our variables, then crosstabs of gender and SES as the exposure and expenditure as the outcome:

```
proc format;/*note use of proc format to create spendfmt and
   format to apply it*/
   value spendfmt 1="High Utilization"
                  0="Low Utilization"
                  ;
run;

proc freq data=expend;
   tables expenditure gender ses age
             gender*expenditure ses*expenditure;
   format expenditure spendfmt.;
run;
```

The next screenshot presents the one-way tables for our first three variables. We can see that SAS has applied our format for expenditure. There do not appear to be any particular problems with these variables, except (again) that SAS's alphanumeric approach to ordering variables is inappropriate and misleading (Fig. 8.5).

In the following screenshot, we see that perhaps age is not the most appropriate variable for a categorical analysis. It does, though, illustrate that PROC FREQ can be a good way to catch miscoded variables, weird entries, etc. (Fig. 8.6).

The following screen presents the results of our gender by outcome cross tabulation (Fig. 8.7):

If your first impression is that SAS presents a lot of data in a small space, you are correct. You should commit to memory that the four numbers in each cell are: frequency, percent of total, row percentage, and column percentage. As in our simple introductory example about weather and mood, we are interested in the row percentages. Here we see that health-care expenditure does seem to vary by gender, with women appearing to spend more (68 % of men were in the low-utilization category compared to 58 % of women).

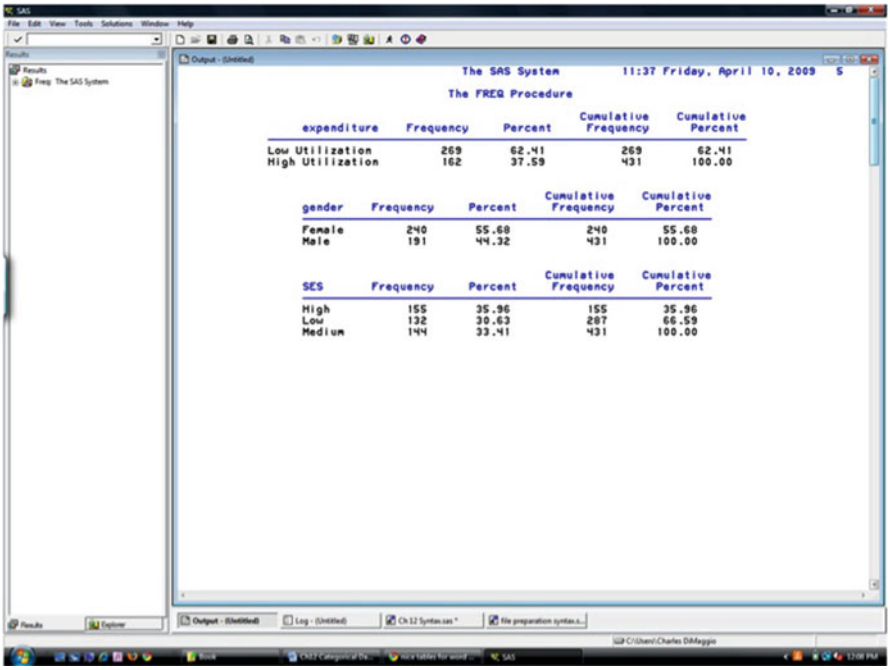


Fig. 8.5 PROC FREQ results screen

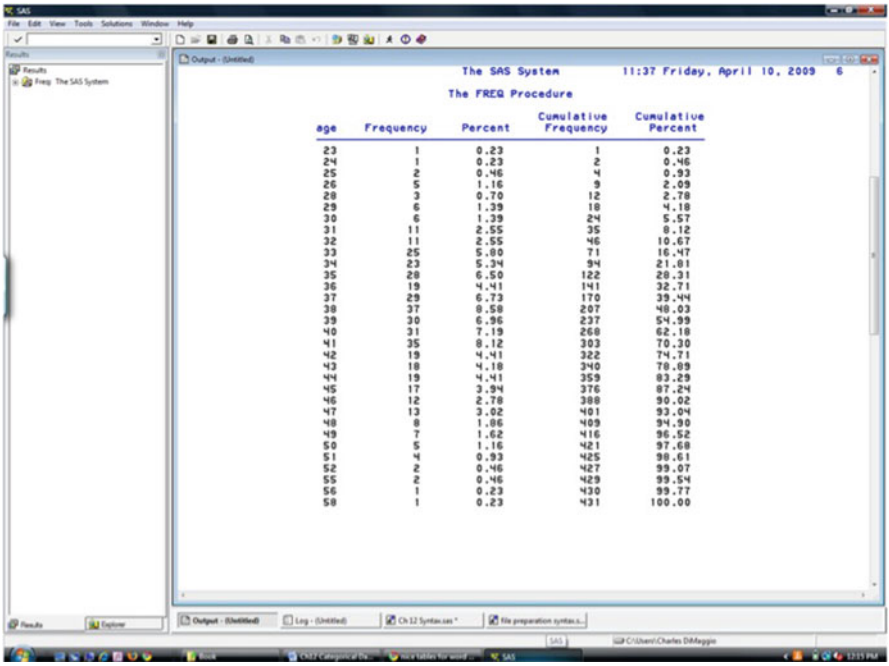
Let's turn our attention to the association between SES and health-care expenditure (Fig. 8.8).

Here (again looking at row percentages) we see that 48 % of high-income folks were in the high-utilization category compared to 32 % of low-income folks. And again, the ordering of the SES categories is problematic. It is, at best, distracting and, at worst (if there is some kind of trend in the data) misleading. We will have to address this.

8.4 Reordering Categorical Variables

To reorder variables for categorical analyses, we use a DATA step to create a data set in which a character variable is transformed into a numeric variable which corresponds to logical the order in which we want the categorical variable[3]:

[3]We are using a neat bit of Boolean logic here, instead of many IF THEN statements. In SAS an expression enclosed in parentheses is a logical operator that returns the value 1 if the expression is true and 0 if it is false. This syntax tells SAS to look at each observation, if the $oldvar = low$, set it to 1 (true) and if not, set it to 0 (false). We do that for each level (med and high) and then multiply



The SAS System 11:37 Friday, April 10, 2009 6

The FREQ Procedure

age	Frequency	Percent	Cumulative Frequency	Cumulative Percent
23	1	0.23	1	0.23
24	1	0.23	2	0.46
25	2	0.46	4	0.93
26	5	1.16	9	2.09
28	3	0.70	12	2.78
29	6	1.39	18	4.18
30	6	1.39	24	5.57
31	11	2.55	35	8.12
32	11	2.55	46	10.67
33	25	5.80	71	16.47
34	23	5.34	94	21.81
35	28	6.50	122	28.31
36	19	4.41	141	32.71
37	29	6.73	170	39.44
38	37	8.58	207	48.03
39	30	6.96	237	54.99
40	31	7.19	268	62.18
41	35	8.12	303	70.30
42	19	4.41	322	74.71
43	18	4.18	340	78.89
44	19	4.41	359	83.29
45	17	3.94	376	87.24
46	12	2.78	388	90.02
47	13	3.02	401	93.04
48	8	1.86	409	94.90
49	7	1.62	416	96.52
50	5	1.16	421	97.68
51	4	0.93	425	98.61
52	2	0.46	427	99.07
55	2	0.46	429	99.54
56	1	0.23	430	99.77
58	1	0.23	431	100.00

Fig. 8.6 PROC FREQ results for a continuous variable

```
data expend;
   set expend;
   SES_level=1*(SES='Low') + 2*(SES='Medium')
            + 3*(SES='High');
run;
```

We then use PROC FORMAT to create a user defined format:

```
proc format;
   value sesfmt 1='Low Expenditure'
                2='Medium Expenditure'
                3='High Expenditure';
run;
```

Finally, we use PROC FREQ on our reordered numeric variable with this FORMAT statement:

```
proc freq data=expend;
   tables SES_level*expenditure;
   format SES_level sesfmt. expenditure spendfmt.;
run;
```

We now see the SES variable output more logically (Fig. 8.9).

---

the 1 or 0 the logical expression returns by 1, 2, or 3 as indicated. There are other ways to do this, but this quite effectively transforms a text variable into a numeric variable.

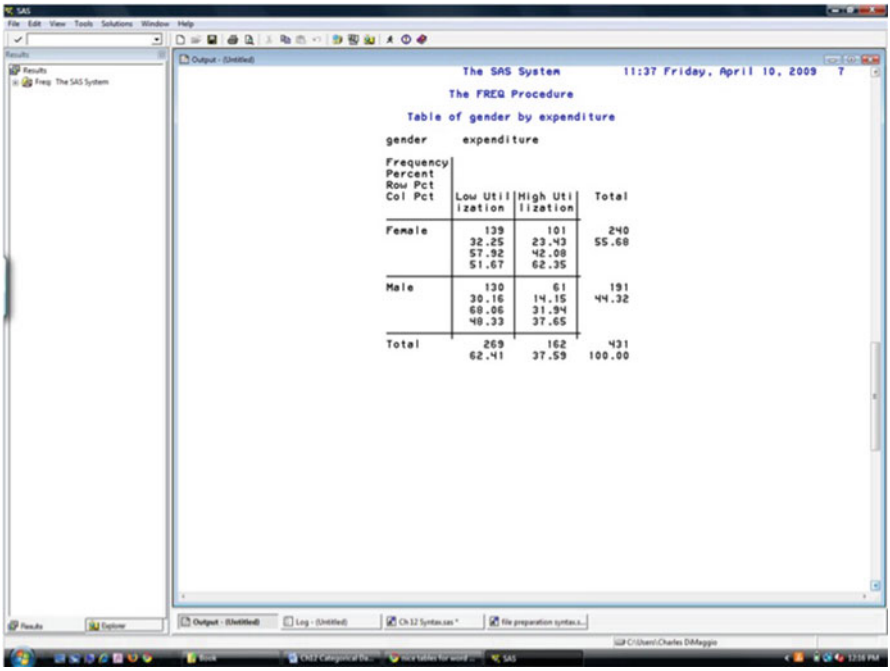


Fig. 8.7 Cross tabulation of dichotomous variables

8.5 Tests of Statistical Significance for Categorical Variables

The following statistical tests are, essentially, assessments of the likelihood of observing our results given a null hypothesis under a statistical model. They are not, with the exception of the Spearman correlation coefficient, assessments of the *strength* of any observed or apparent association.

8.5.1 Chi-Square

We will, at a minimum, want a measure of whether possible associations, as evidenced by differing values in rows, are due to chance. For such categorical data, the chi-square (χ^2) statistic is commonly used for this purpose.[4]

[4]Perhaps surprisingly, given its almost universal use for categorical data, chi-square has an underlying assumption of normality. This is why there are, in fact, sample size requirements. We need enough data for the approximation of the normal distribution.

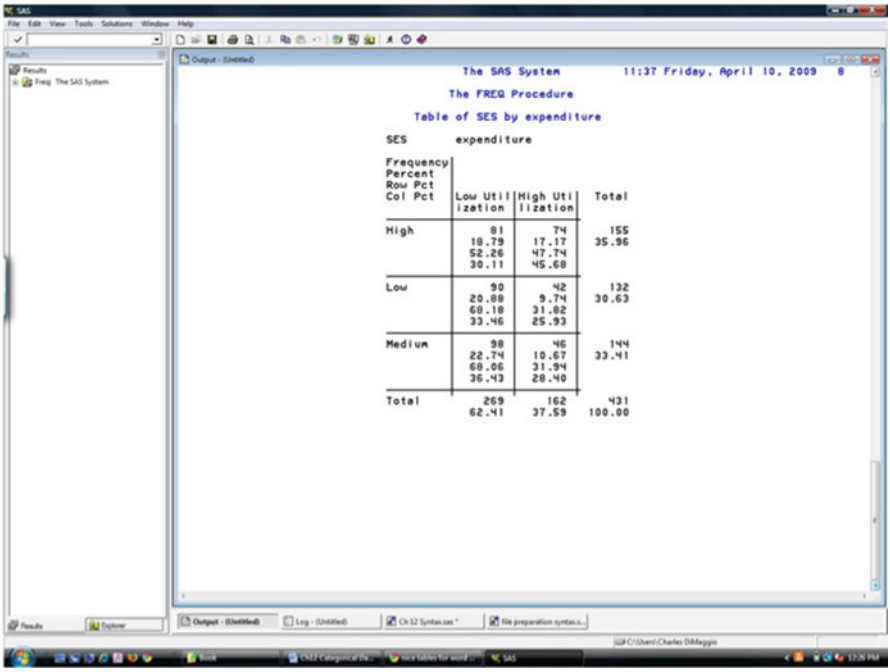


Fig. 8.8 Cross tabulation of categorical variables with more than two levels

χ^2 is used to assess the statistical significance of an association among nominal variables and is calculated as the sum of the squared differences between observed and expected values divided by the expected value:[5]

$$\chi^2 = \frac{\Sigma(O-E)^2}{E}, \tag{8.1}$$

where E is the expected value and is equal to the $rowtotal * columntotal / grandtotal$[6] and O is the observed value. Our null hypothesis is that our observed values are equal to our expected values. The p-value for a χ^2 is the probability of observing a test statistic at least as large that we obtain under our null hypothesis.

In SAS χ^2 is invoked as an option after the TABLES statement in PROC FREQ, for example,

```
TABLES var_a*var_b / CHISQ
```

[5]Note that there is a chi-square statistic called the Mantel–Haenszel chi-square specifically for ordinal data.

[6]This is derived from column total divided by the grand total, which gives the expected proportion, which is then multiplied by the row total to get the expected cell count.

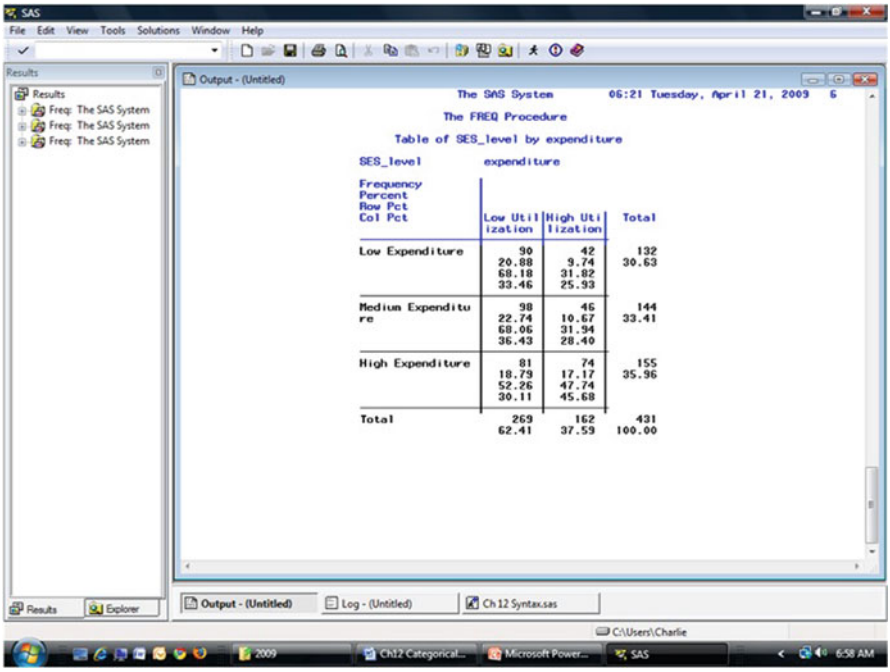


Fig. 8.9 Categorical variables reordered

Some helpful options after the CHISQ include listing the expected frequency in a cell ("expected") and instructions not to print column totals or percentages ("nocol," "nopercent").

Let's use the "expend" data set to write a statement using PROC FREQ invoking a χ^2 statistic to test the association between gender and health-care expenditure. We will request the expected numbers and use the "nocol" and the "nopercent" options to suppress column percentages and cell percentages:

```
/* expenditure chi square exercise */

proc freq data=expend;
   tables gender*expenditure
          / chisq expected nocol nopercent;
run;
```

We can see that there is indeed a statistically significant association between gender and the expenditure with a p-value of 0.03 (Fig. 8.10).

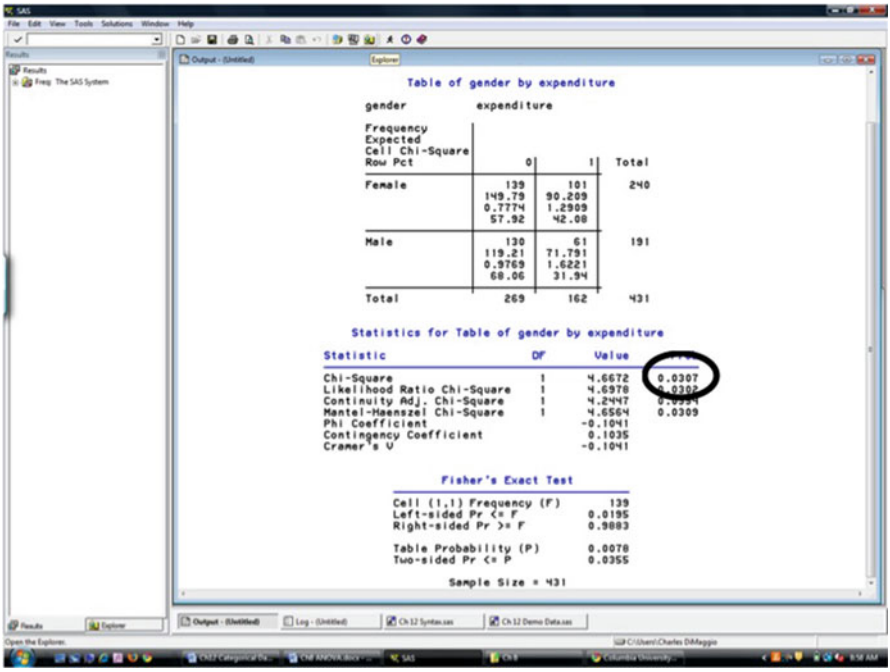


Fig. 8.10 Statistics for tabular data

DON'T BE SQUARE

χ^2 is *not* a measure of the strength of an association. It only measures the statistical significance of an association. It is as much a reflection of the sample size as of any underlying effect or association. If we double the numbers in the cells, we double the chi-square. A couple of more salient points about χ^2:

- It is used for frequencies or counts not percentages or proportions.
- For it to be an appropriate statistic, there must be sufficient numbers in the table cells. Cochrane's rule is that 80 % of the cell entries should be greater than 5, and no cells should have cell counts less than 1. For a 2×2 table, none of the cells should have expected counts less than 5.

8.5.2 Exact Tests

As noted above, χ^2 is only appropriate where there are sufficient numbers in the table cells. To address the situation when this is not the case we have recourse

to so-called "exact" tests. You will see reference to two flavors of exact tests. The Fisher's exact test is based on the hypergeometric distribution and is intended for a 2×2 table where the marginal values are fixed. This is the form perhaps most commonly encountered in epidemiology. SAS also provides for the so-called Pearson chi-square statistics which, as its name implies, is based on the chi-square distribution. It may be used for any $n \times n$ table.

Fisher's exact test is indicated for situation where one of the cells in a 2×2 contingency table has fewer than the recommended count of 5. In SAS, Fisher's exact test is invoked as an option under a "tables" statement:

```
tables varA*varB / fisher;
```

The more general Pearson chi-square is invoked with an exact statement:

```
exact pchi;
```

The formula for the calculation of Fisher's exact test is

$$r_1!r_2!c_1!c_2!/n!f_{11}!f_{12}!f_{21}!f_{22}!, \tag{8.2}$$

where r and c are the row and column totals, n is the grand total and f_{rc} is the cell frequency.

The calculation involves the following four steps:

1. Lay out the table.
2. Lay out all the more extreme tables.
3. Calculate the χ^2 probability for each small frequency 2×2 table.
4. Sum all of these probabilities.

As an example, consider the following 2×2 table:

	Disease	No disease	
Exposed	0	3	**3**
Not exposed	2	2	**4**
	2	**5**	**7**

To assess the statistical significance of this table, we start with the key assumption that the column and row totals are fixed. We then calculate a χ^2 statistic for each table that could give rise to those fixed marginal values and add up the individual p-values. In this example, the three tables that could have given rise to the above table are, first, the observed table

and two hypothetical but potential tables:

Notice again how our choice of tables is constrained the marginal totals.

	Disease	No disease	
Exposed	0	3	**3**
Not exposed	2	2	**4**
	2	**5**	**7**

	Disease	No disease	
Exposed	1	2	**3**
Not exposed	1	3	**4**
	2	**5**	**7**

	Disease	No disease	
Exposed	2	1	**3**
Not exposed	0	4	**4**
	2	**5**	**7**

For each of the three possible tables, we calculate a χ^2 statistic and its associated p-value. Here, they are

	χ^2	p-value
Observed table	2.100	0.286
Possible table 1	0.058	0.571
Possible table 2	3.733	0.143

Lastly, we add up the p-values: $0.286 + 0.058 + 0.143 = 0.487$.

SAS, as usual, makes short work of this process for us. As a demonstration, let's read in the following small data set:

```
data pexact;
  input a b ;
  datalines;
1 2
1 2
1 2
2 1
2 1
2 2
2 2
;
run;
```

Let's take a look at the 2×2 contingency table for the variable a vs. b (Fig. 8.11).

Clearly, there are insufficient numbers to justify the use of a χ^2 statistic. Let's request a Fisher's exact test:

```
proc freq data=exact;
   tables a*b;
   exact pchi;
run;
```

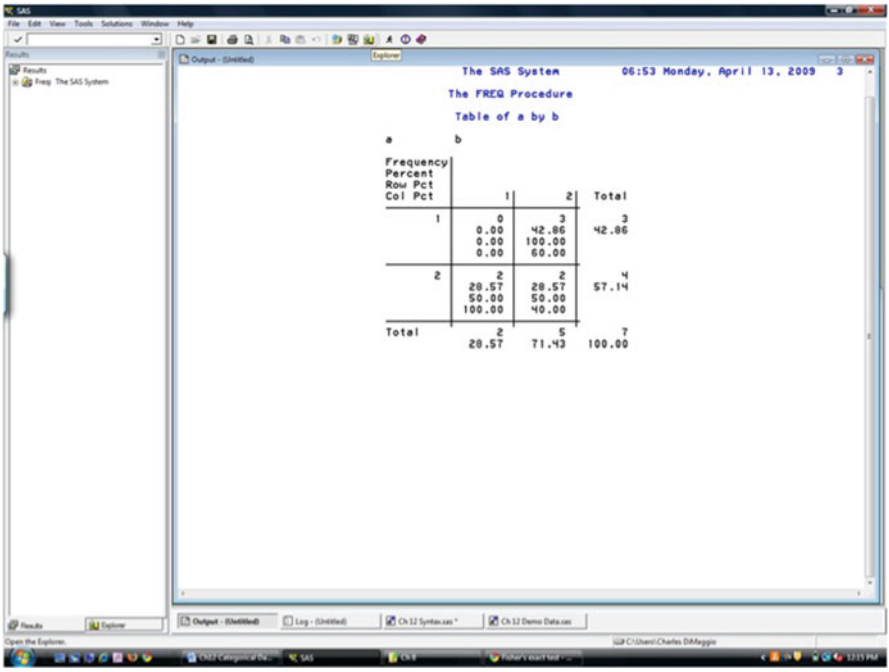


Fig. 8.11 PROC FREQ result for a table with small numbers

Note that on the first screen, we get a warning that χ^2 is not a valid test (Fig. 8.12).

The next screen presents the results of the Fisher's exact test (Fig. 8.13).

The p-value associated with the exact test is 0.4 – clearly not statistically significant. Note, though, that SAS also presents the usual "asymptotic"[7] χ^2 statistic.

EXACTLY

Note that depending on your software version, the SAS request for the usual χ^2 statistic may also return a default "Exact" statistic. It may or may not be the one you want. Requesting the specific statistic appropriate to your data will always work.

[7]Asymptotic refers to an approximation to the large-number normal distribution. It is one of those great scrabble words, like heteroscedasticity, that tend to arise in statistics. Amazon.com has a feature called Statistically Improbable Phrases or SIPs that pops up when you search for statistics textbooks.

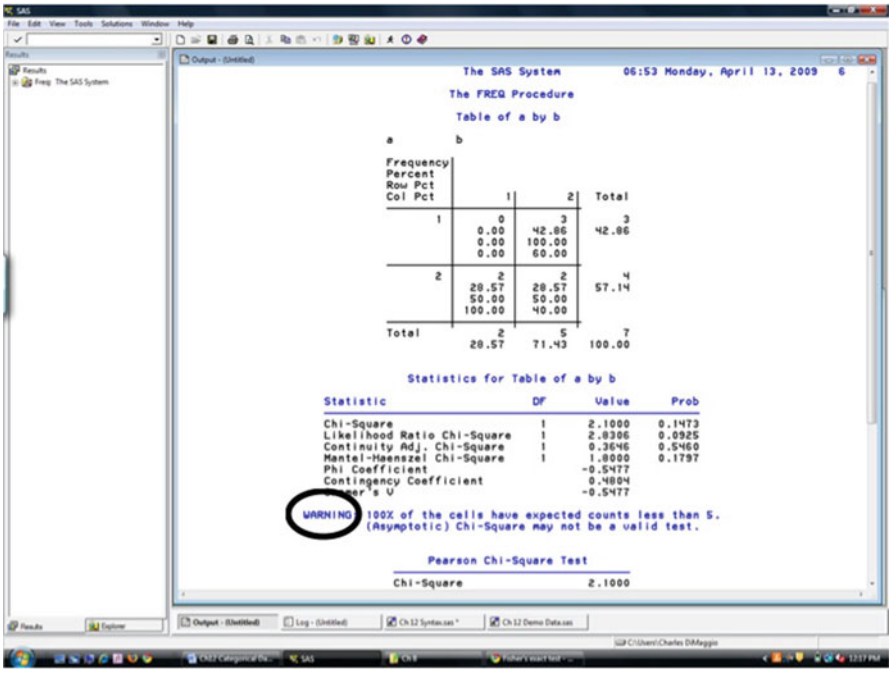


Fig. 8.12 Statistics for a table with small numbers

8.5.3 The Mantel–Haenszel Chi-Square

When we are interested in assessing the statistical significance of *ordinal* associations, a more powerful test statistic is the Mantel–Haenszel chi-square[8] (sometimes referred to as "chi-square for trend"). It is more powerful in the setting of a linear trend because it is restricted to just that one situation, as opposed to the general χ^2 which is intended for all possible associations.

But, be careful, the test assumes a linear association and can be thrown off because it looks at the average trend. If there is no underlying linear trend, it can give you the wrong answer.

There is an additional statistic in the SAS armamentarium called the *mean score statistic*, which is recommended for nominal by ordinal associations. Again, the usual and customary χ^2 is the more conservative approach. In either setting, you can't go too far wrong by just relying the plain vanilla χ^2.

[8]No, this is not the Mantel–Haenszel *Odds Ratio* with which epidemiologists are most familiar. Mantel and Haenszel were busy folks. We'll get to that, Mantel–Haenszel, which SAS refers to as the Cochran–Mantel–Haenszel for stratified 2×2 tables, in the next chapter.

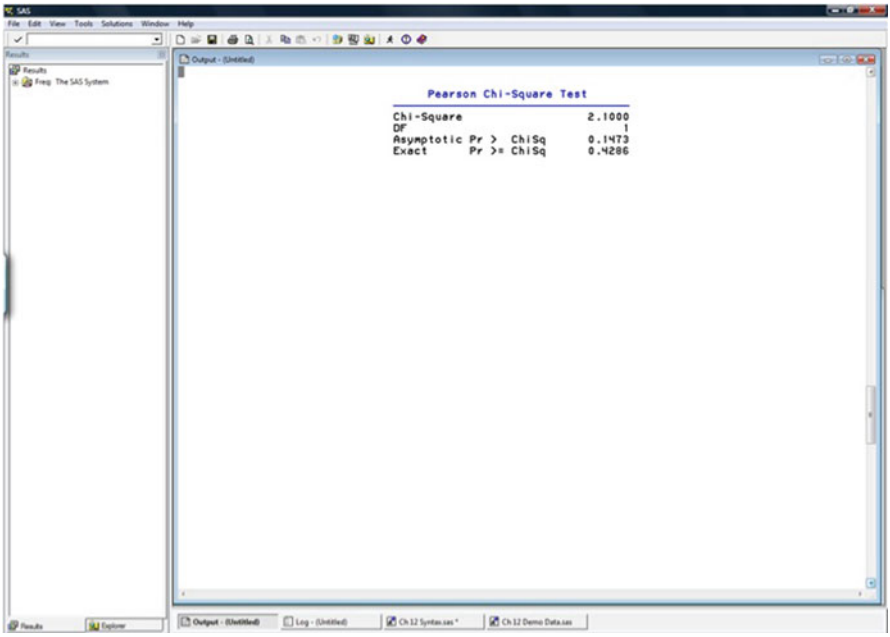


Fig. 8.13 Exact test results for a table with small numbers

8.5.4 The Spearman Correlation Coefficient

So far we have just considered the *statistical significance* of an association. If we want to measure the *strength* of an association (and as epidemiologists we generally do), other tests are required. One of these is the Spearman correlation coefficient.

The Spearman correlation statistic is a nonparametric cousin of the Pearson correlation coefficient, which we will consider in a future chapter, which uses the ranks of data. Since it is nonparametric and so by definition doesn't require the parametric assumptions of normality, etc., it can be used in the setting of categorical data. Like its Pearson analog, the Spearman statistic has a range between -1 and 1, so it gives us a measure of the strength of the association.

The Spearman statistic is invoked with the MEASURES statement as an option under PROC FREQ. Adding CL will return confidence limits for the statistic.

As an example, let's return to our health-care expenditure data. We will rerun the PROC FREQ syntax looking at the association between SES and health care expenditure. Only this time, we will request measures of both statistical significance (χ^2) and the strength of any association (Spearman correlation coefficient) (Fig. 8.14):

```
proc freq data=expend;
   tables SES_level*expenditure / chisq measures cl;;
   format SES_level sesfmt. expenditure spendfmt.;
run;
```

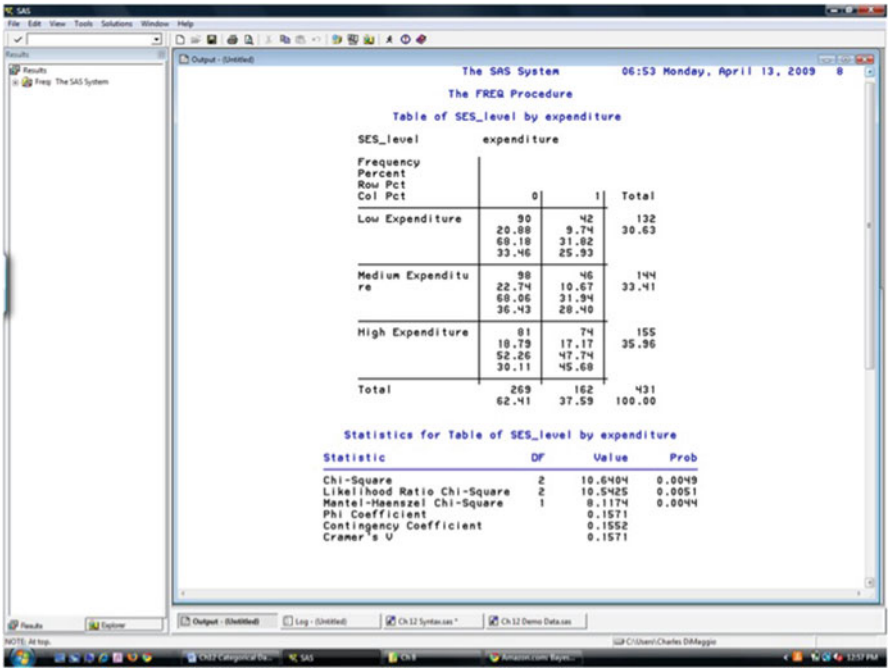


Fig. 8.14 Statistics for the association of categorical variables (1)

We might suspect that there is an underlying linear trend, and because the Mantel–Haenszel χ^2 is significant, we can conclude there is evidence of an ordinal association between purchase and income. It is, though, important to note that the statistic tells us nothing of the strength of any correlation. For that, we look at the Spearman correlation statistic and its 95 % confidence bounds (Fig. 8.15).

Our results indicate a relatively small (0.14) positive ordinal association. The "ASE" in this table refers to the asymptotic standard error or what our standard error would approach as sample size approaches infinity. The confidence limits are valid as long as our sample size is large enough. The rule of thumb is we need at least 25 data points for each degree of freedom.

8.6 Significance vs. Strength

To illustrate the difference between statistical significance and the strength of an association, let's take the health-care expenditure data set we've been working with in this chapter, and just double it to create the data set "expend2." Notice, all we are doing is cloning and appending the same data set to itself:

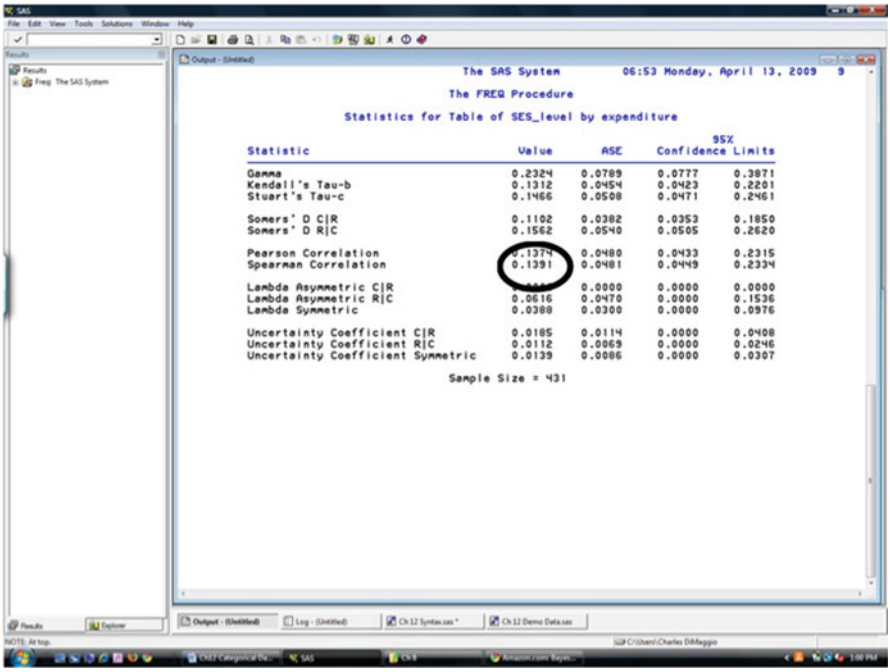


Fig. 8.15 Statistics for the association of categorical variables (2)

```
data expend1;
set expend;
run;

data expend2;
set expend expend1;
run;
```

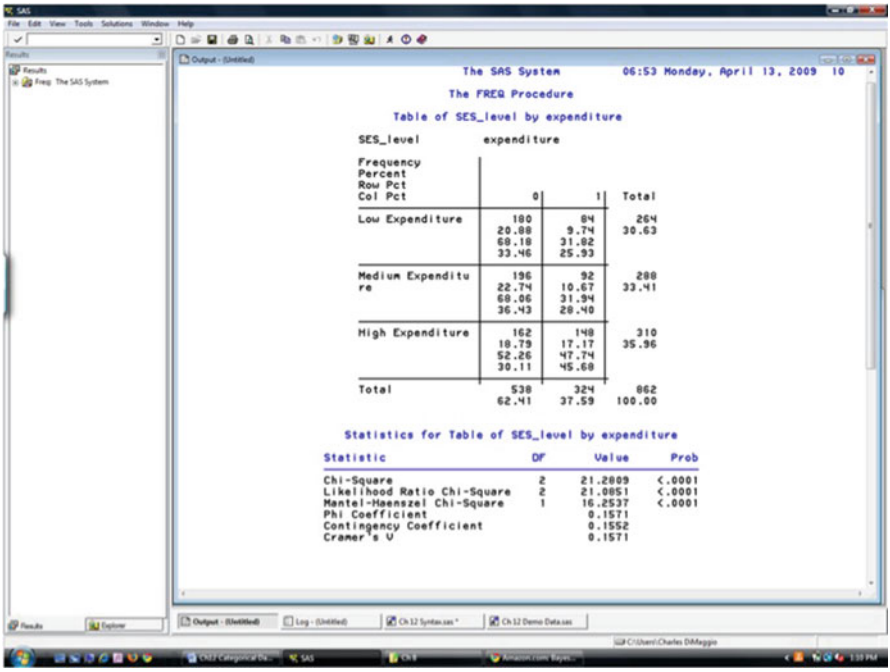
Now, let's rerun the syntax for the significance and strength of the ordinal association between SES and expenditure:

```
proc freq data=expend2;
   tables SES_level*expenditure / chisq measures cl;;
   format SES_level sesfmt. purchase spendfmt.;
run;
```

Let's see how doubling the number of observations affects our test of statistical significance (Fig. 8.16).

Everything is much more "significant." As opposed to a p-value for the Mantel Haenszel χ^2 of 0.004, it is now less than 0.0001, which in SAS-talk means very, very, very small. What about the test of the strength of association (Fig. 8.17)?

We see that it remains 0.14 (although we see that the confidence limit is tighter).



The SAS System 06:53 Monday, April 13, 2009 10

The FREQ Procedure

Table of SES_level by expenditure

SES_level expenditure

Frequency Percent Row Pct Col Pct	0	1	Total
Low Expenditure	180 20.88 68.18 33.46	84 9.74 31.82 25.93	264 30.63
Medium Expenditure	196 22.74 68.06 36.43	92 10.67 31.94 28.40	288 33.41
High Expenditure	162 18.79 52.26 30.11	148 17.17 47.74 45.68	310 35.96
Total	538 62.41	324 37.59	862 100.00

Statistics for Table of SES_level by expenditure

Statistic	DF	Value	Prob
Chi-Square	2	21.2809	<.0001
Likelihood Ratio Chi-Square	2	21.0851	<.0001
Mantel-Haenszel Chi-Square	1	16.2537	<.0001
Phi Coefficient		0.1571	
Contingency Coefficient		0.1552	
Cramer's V		0.1571	

Fig. 8.16 The effect of an increased sample size on statistical significance (1)

Again, it is important to remember that tests of statistical significance are as much reflections of sample size as anything else. In the next chapter, we will spend more time with measures that give us a sense of both the significance and the strength of an association.

Problems

You are an injury epidemiologist interested in vehicle safety as it relates to several other variables. Your measure of vehicle safety is based on a score given to vehicle models using the frequency of insurance claims.

The text file "cars" contains the following variables:

- Safety—safety score (1=below average, 0=average or above)
- Type—type of vehicle (sports, small, medium, large, and sport/utility)
- Region—manufacturing region (Asia, N America)
- Weight—weight of the vehicle in thousands of pounds

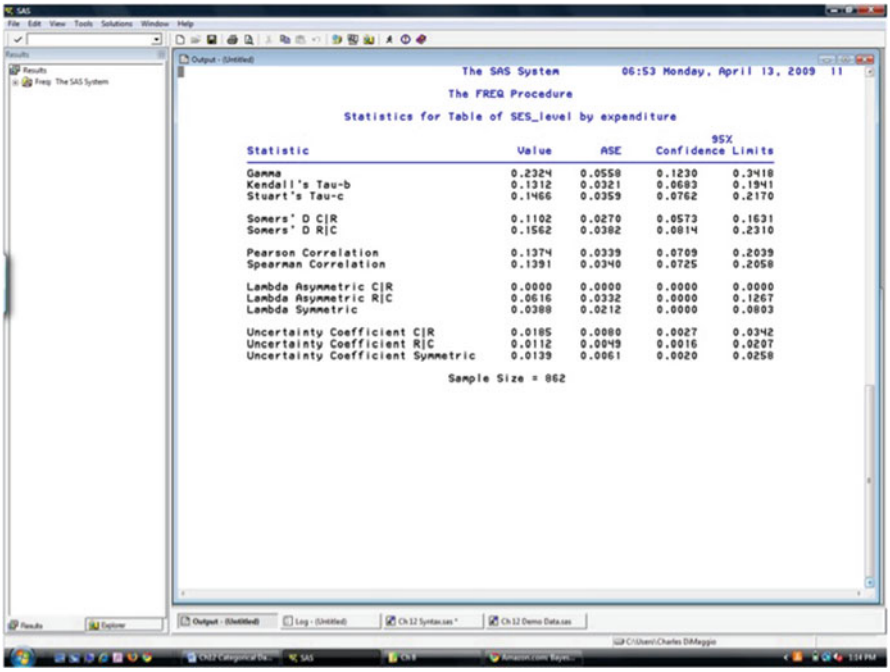


Fig. 8.17 The effect of an increased sample size on statistical significance (2)

8.1. One-Way Frequencies

- Read the file into SAS and use the PRINT procedure to examine data. Provide the first 20 observations from the PRINT procedure.
- Use PROC FREQ to create one-way frequency tables for the variables safety, type, and region.
- What is the measurement scale of each variable (safety, type, region, weight)?
- What is the proportion of cars made in North America?
- For the variables safety, type, and region, are there any unusual data values that warrant further investigation?

8.2. Cross Tabulations

Use PROC FREQ to examine the cross tabulation of region by safety. Generate a temporary format to clearly identify the values of safety. Along with the default output, generate the expected frequencies and the chi-square:

- For the cars made in Asia, what percentage had a below-average safety score?
- For the cars with an average or above safety score, what percentage was made in North America?
- Do you see any association between region and safety?
- What cell contributed the most to any possible association?

8.3. Chi Square

Perform a chi-square test of the statistical significance of any association between region and safety. Interpret the p-value from the test with respect to probability. Do you reject or fail to reject the null hypothesis at the 0.05 level?

8.4. Spearman

Create a new variable named size by assigning 1 for type equal to small or sports, 2 for type equal to medium, and 3 for type equal to large or sport/utility. Examine the ordinal association between size and safety using PROC FREQ:

- What statistic should you use to detect an ordinal association between size and safety?
- Do you reject or fail to reject the null hypothesis at the 0.05 level?
- What is the strength of the ordinal association between size and safety?
- What is the 95 % confidence interval around that statistic?

Chapter 9
Categorical Data Analysis II

Abstract At the end of the previous chapter, we spent some time considering the difference between statistical significance and the strength of an association. In this chapter we spend some time on a measure that combines elements of both: the Odds Ratio, which for 2×2 tables is the recommended test of both the strength and the statistical significance of an association. As with many SAS statistical procedures we have seen, the actual execution of the procedure is fairly simple. The challenge is in understanding the underlying statistical process and interpreting the results.

9.1 Probabilities and Odds

In preparation for a discussion of the odds ratio we must first set the stage with a brief consideration of probabilities and odds.

A *probability* is a number from 0 to 1 inclusive, usually expressed in fraction form. It is the ratio of the number of chances of a specific event to the total number of chances possible. For example, if I have four marbles in a jar, three red and one blue, then the probability of drawing the blue marble is 1/4. There is one chance of a blue marble out of four total chances (marbles).

Odds are expressed as the number of chances for (or against) vs. the number of chances against (or for) some event occurring. So, in the above marble example, since there is one chance of your picking the blue marble and three chances of your picking a red marble, the odds are 3 to 1 against you picking the blue. For odds in favor, we just reverse them. The odds are 1 to 3 in favor of you picking the blue marble.[1]

There is a one-to-one correspondence between probabilities and odds. To convert odds to probability, we add up the total chances, i.e., sum the odds to represent the total number of chances of an event occurring, and use that as the denominator for

[1]Note that this does not mean that the probability is 1/3 for or against.

C. DiMaggio, *SAS for Epidemiologists: Applications and Methods*,
DOI 10.1007/978-1-4614-4854-9_9,

the probability. So, if the odds against a horse winning are 4 to 1, this means that, out of 5 (4 + 1) total chances, the horse has 1 chance of winning. The probability of the horse winning is 1/5 or 20 percent. To convert a probability to odds we remove the event from the total chances and set them against each other as a ratio.

The calculations are actually quite simple:

```
odds = probability / 1-probability
probability = odds / 1+odds
```

When the probability is 50%, we say it is even odds (1:1). When the probability of an event is larger than 50%, then the odds for the event will be larger than 1.

9.2 The Odds Ratio

9.2.1 Why Epidemiologists Need the Odds Ratio

For epidemiologists, the odds ratio was a response to the need for a measure of association that could be used with case-control data.

Many epidemiological data are amenable to the calculation of rate ratios and risk ratios. While they are quite different conceptually (one uses rates, the other probabilities), they are both predicated on the idea that we know how much disease occurs in an exposed population vs. an unexposed population. Crucially, the "direction" is going from exposure to disease in that we start off knowing the exposure status in non-diseased individuals.

In case-control studies, this assumption is turned on its head; we now start off knowing the disease status of the participants and go backwards to figure out the exposure status. In terms of conditional probabilities, forward-looking study designs like cohort studies seek to determine the probability of disease given exposure:

$$Cohort\ studies : Prob[disease|exposure]. \tag{9.1}$$

Case-control studies, on the other hand, can only directly present data on the probability of exposure given disease:

$$Case\text{-}control : Prob[exposure|disease]. \tag{9.2}$$

Very importantly, we cannot directly calculate the risk or probability of disease (risk ratios or rate ratios) because we don't know the original population at risk. If, for example, we are interested in cigarette smoking as an exposure and lung cancer as an outcome, we could contrast these two approaches as

$$Cohort\ studies : P[disease|exposure] = P[lung\ cancer|smoking] \tag{9.3}$$

$$Case\text{-}control : P[exposure|disease] = P[smoking|lung\ cancer]. \tag{9.4}$$

Let's look at a simple example that illustrates how very different these two probabilities actually are:

	Lung cancer	No lung cancer	
Smoker	100	900	1,000
Nonsmoker	50	1,950	2,000

The two calculations would be:

$$Cohort : P[disease|exposure] = 100/1,000 = 0.10 \tag{9.5}$$

$$Case\text{-}Control : P[exposure|disease] = 100/150 = 0.66. \tag{9.6}$$

The inability to get at risk or probability of disease with case control data limited the effectiveness of the case-control approach in epidemiology and motivated the search for the measure of effect that would be valid for estimating risk (P[D|E]) using case-control (P[E|D]) data. That measure of effect is the odds ratio.

As we noted, there is a simple one-to-one relationship between a probability and its associated odds:

$$odds = probability/1 - probability. \tag{9.7}$$

Or, in terms of the probability of a disease occurring:

$$ODD = P[disease]/1 - P[disease] = P[disease]/P[nodisease]. \tag{9.8}$$

An odds ratio is (logically enough) the ratio of two odds.

This is information we can get from a case-control study.

9.2.2 The Disease Odds Ratio

Say our lung cancer data above came from a *cohort* study. We could easily calculate the *risk* of disease by comparing the outcomes in the exposed vs. the unexposed:

$$(100/1,000)/(50/2,000) = 0.1/0.025 = 4. \tag{9.9}$$

What is the odds ratio for this cohort study? It is the odds of disease given exposure over the odds of disease given no exposure. In terms of probabilities, this is

$$\frac{(P[disease|exposure]/P[no\ disease|exposure])}{(P[disease|no\ exposure]/P[no\ disease|no\ exposure])}. \tag{9.10}$$

We can get all this information from our cohort study. We begin with the odds of disease given exposure:

$$P[disease|exposure]/P[no\ disease|exposure] \tag{9.11}$$
$$= P[disease|exposure]/1 - P[disease|exposure] \tag{9.12}$$
$$= (a/a+b)/(b/a+b) \tag{9.13}$$
$$= (100/1,000)/(900/1,000) \tag{9.14}$$
$$= 0.1/0.9 = 0.11. \tag{9.15}$$

We can similarly get at the odds of disease given no exposure:

$$P[disease|no\ exposure]/P[no\ disease|no\ exposure] \tag{9.16}$$
$$= P[disease|no\ exposure]/1 - P[disease|no\ exposure] \tag{9.17}$$
$$= (c/c+d)/(d/c+d) \tag{9.18}$$
$$= (50/2,000)/(1,950/2,000) \tag{9.19}$$
$$= 0.025/0.927 = 0.026. \tag{9.20}$$

The ratio of these two odds is our disease odds ratio:
$DiseaseOR = 0.11/0.026 = 4.3$.
Note how close this is to our risk ratio.[2]

Assigning the usual a,b,c,d letters to a 2×2 epidemiological Punnett square, the disease odds ratio can be summarized as:

$$disease\ odds\ ratio = \frac{[(a/a+b)/(b/a+b)]}{[(c/c+d)/(d/c+d)]} = \frac{(a/b)}{(c/d)} = \frac{ad}{bc}. \tag{9.21}$$

Certainly knowing the odds ratio is a good approximation of the relative risk in cohort studies is nice enough, but how does it help us with case-control data? The answer lies in the exposure odds ratio.

[2]We used to teach the "rare disease assumption" as the reason that a case-control study can approximate a cohort study. There are now more fundamentally sound reasons supporting the validity of case-control studies. Rothman, in particular, has a very nice discussion. Still, the reason the odds ratio is a valid approximation to the relative risk is because the outcomes we study are, in fact, rare and do not when removed detract appreciably from the bottom number of the ratios themselves when compared to the denominator of the analogous probability. (See the formula for converting probabilities to odds.) Odds ratios will overestimate risk, when the outcomes are common. As a general rule of thumb, I find outcomes in much beyond 10 to 15 percent of the study population to be problematic.

9.2.3 The Exposure Odds Ratio

Now let's say our lung cancer data is from a case-control study that arose from the same cohort (all case-control studies are, in fact, based on some underlying cohort...) and got the same results:

	Lung cancer	No lung cancer	
Smoker	100	900	?
Nonsmoker	50	1,950	?

Notice, though, that we do not know the exposure experience of the underlying data from which the cases and controls arose. Now, we can't get the odds of disease given exposure vs. the odds of disease given no exposure (what we really want). But, but *can* get at the odds of exposure given disease vs. the odds exposure given no disease:

$$P[exposure|disease]/P[noexposure|disease] \tag{9.22}$$
$$= P[exposure|disease]/1 - P[exposure|disease] \tag{9.23}$$
$$= (a/a+c)/(c/a+c) \tag{9.24}$$
$$= (100/150)/(50/150) = 2. \tag{9.25}$$

Similarly, we can get the odds of exposure given no disease:

$$P[exposure|no\ disease]/P[no\ exposure|no\ disease] \tag{9.26}$$
$$P[exposure|no\ disease]/1 - P[exposure|no\ disease] \tag{9.27}$$
$$(b/b+d)/(d/d+c) \tag{9.28}$$
$$(900/2,850)/(1,950/2,850) = 0.46. \tag{9.29}$$

So, our exposure odds ratio is $= 2/0.46 = 4.3$.

This is, in its own small way, an exciting development. We see that it is the exact same result as the disease odds ratio from the cohort data. And, in fact, if we were to summarize the exposure odds ratio as we did the disease odds ratio in terms of an a,b,c,d 2×2 Punnett square, our result would be:

$$exposure\ OR = \frac{[(a/a+c)/(c/a+c)]}{[(b/b+d)/(d/d+c)]} = \frac{(a/c)}{(b/d)} = \frac{a/d}{b/c}. \tag{9.30}$$

In fact the calculation for the odds ratio, under any study setting is:

$$OR = \frac{ad}{bc}. \tag{9.31}$$

The odds ratio is a measure of association that is invariant across study design, and allows us to approximate prospective results, from retrospective data.

It is easy to get an odds ratio in SAS by requesting MEASURES under PROC FREQ in the setting of a 2×2 table.

9.3 Preterm Labor and Birth Weight Example 1

We will demonstrate an odds ratio calculation in SAS with a data set of 189 observations examining at the relationship between previous preterm labor and subsequent low birth weight.

We begin by looking at the result of our nlevel table, making sure there is the correct number of levels for each variable and that there are no zero cells in our 2×2 table (Fig. 9.1):

```
proc freq data=birth nlevels;/* nlevels num levels for
  variables */
tables prev_pretrm*low / measures; /* 2x2 table measures
  request OR */
title 'odds ratio for association between previous preterm and
  lbw';
run;
```

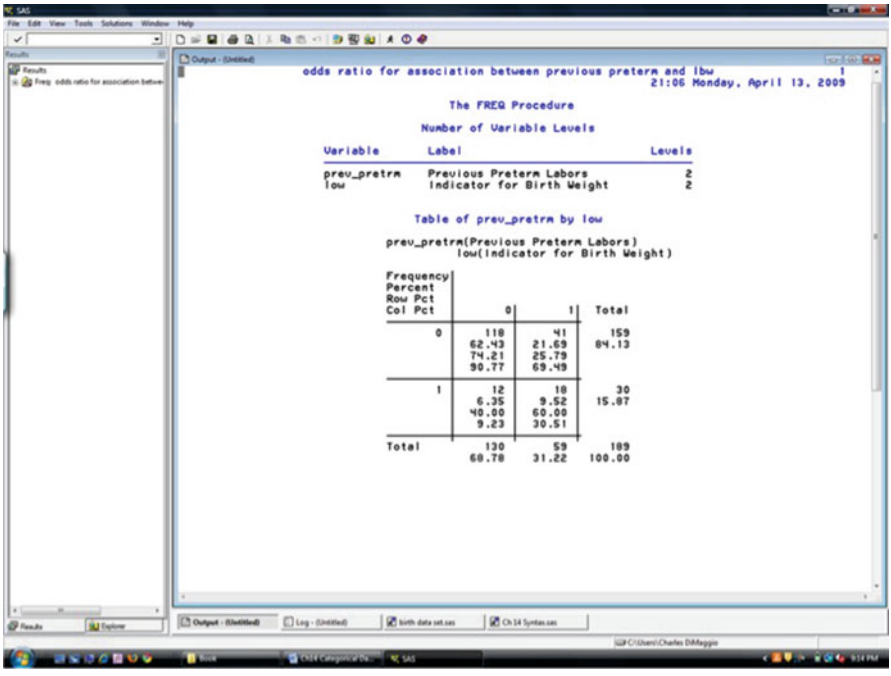


Fig. 9.1 A 2×2 table

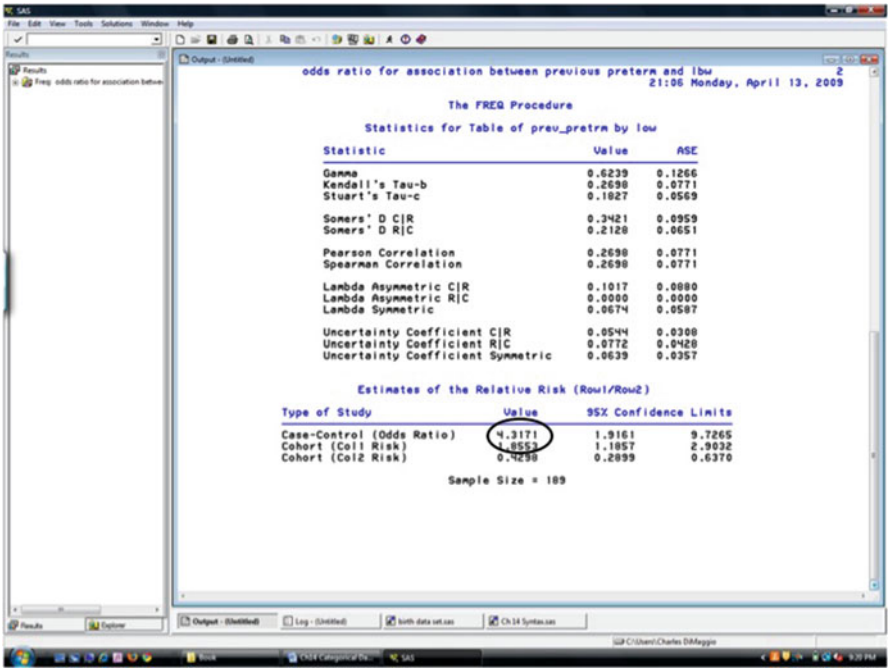


Fig. 9.2 An odds ratio for a 2×2 table

We note (not shown) that the chi-square result indicates a statistically significant association between previous preterm labor and low birth weight. Since this is a 2×2 table, we can turn to the odds ratio calculation, where we see a strong ($4\times$) measure of association (Fig. 9.2)

ALL TOO MUCH

By way of review…

- R×C tables: Spearman correlation
- 2×2 tables: Odds Ratio
- Ordinal X Ordinal: M-H for association, Spearman for strength
- Nominal X Ordinal: mean score stat for association, uncertainty coeff for strength
- Nominal X Nominal: Pearson chi sqr for association, uncertainty coefficient for strength

9.4 Confounding

Epidemiology is much concerned with confounding. Methodological thinking about what constitutes confounding has evolved over time. Traditionally, a confounder has been viewed as a variable that (a) is a cause of the disease under study (b) is associated with the exposure under study and (c) is not a mediator of the exposure under study. Confounding is usually considered something that arises from the nature of the relationships of the variables, rather than an error in how a study is conducted or how data is collected, which are grouped under the heading of bias.

Hennekens makes the point that the confounder need not be an actual cause of the disease and may be a covariate of the actual causal factor. He also notes that a confounder must not be associated with the disease only among the exposed. Its association with the disease must be independent of exposure, i.e., it is causally related among both exposed and unexposed. In an earlier classic text, Schlesselman defined confounding as the effect of an extraneous variable that wholly or partially accounts for the apparent effect of exposure on disease, i.e., any apparent association may be due to some third variable.

Confounding is perhaps easiest appreciated through an example. Kelsey mentions a classic example of confounding from the breast-cancer literature. Early epidemiologic studies suggested that women with breast cancer were less likely to have breast-fed their children, and to have breast-fed for shorter periods of time than controls of the same age. Does this suggest breast-feeding is somehow protective of breast cancer? It certainly does. Further analysis, though, revealed that more breast-feeding and longer periods of breast feeding are associated with having more children. Once this association with parity is controlled for, the apparent association with breast-feeding and breast cancer disappears. The story actually didn/t end there; even further analysis showed that having more children was associated with having a first child at a relatively young age; once this was taken into account, the apparent association with parity disappeared. The actual confounder was age at first full-term pregnancy.

9.4.1 Identifying and Controlling Confounding

If there is, in fact, some third variable accounting for an apparent effect of our exposure variable on our outcome variable, we will want to identify and account for its effects. Classically, comparing the unadjusted (or crude) measure of effect to the adjusted (or stratified) measure of effect is the key to assessing for confounding. This is best illustrated by an example.

Consider the following data on the effect of coffee drinking on myocardial infarction:

	MI	No MI
Coffee	90	60
No coffee	60	90

The odds ratio is $ad/bc = (90)(90)/(60)(60) = 2.25$. Does this indicate an apparent causal effect of coffee drinking on myocardial infarction? In addition to the critical questions of temporality (did coffee drinking precede MI or perhaps folks who have survived an MI tend to drink coffee?) biological plausibility, existing literature on the subject, etc., we must ask: Are there any known risk factors for myocardial infarction that might be associated with drinking coffee? The answer is yes: smoking.[3] It meets the a priori definition of a confounder in that it is causally related to the disease, associated with the exposure, and not a mediator of the exposure disease relationship. Lets look at the data for evidence of a confounding effect.

As noted above, the key to assessing for confounding is stratified analysis. We "control" for a possible confounder, or account for its effects, by looking at the data within homogenous groups of people with and without the confounder. The idea is that if the effect remains among just smokers and just among just non-smokers, then the effect is probably real. If we find the effect is lessened or attenuated depending on whether someone has the potential confounder or not, it may be the confounder that is causing the effect.

In the following two stratified tables, which separates the study participants into strata or levels restricted to those with and without the potential confounder, we do just that. This first table is restricted to *smokers*:

	MI	No MI	
Coffee	80	40	
No coffee	20	10	OR = 1.0

Now, the table for *nonsmokers*

	MI	No MI	
Coffee	10	20	
No coffee	40	80	OR = 1.0

The apparent association of coffee with myocardial infarction disappears when we look at homogenous groups of our putative confounder (smoking).

[3]Smoking is so frequently a confounder that you might want to at least consider including it in almost any study in which it is not the actual exposure of interest.

9.5 Controlling for Confounding

Epidemiologists control for confounding in one of two main ways: through study design or through analytic technique. Each approach has its advantages and disadvantages, which we won't go into here, but it helps to have a general overview.

9.5.1 Controlling Confounding in Study Design

Randomization, as classically seen in randomized clinical trials, is the most effective means of controlling for both known and unknown confounders. A properly randomized sample should assure that there are the same number of people with and without any known (or unknown) confounders in each group. In the above example, we would randomly select study members to either drink coffee or not drink coffee and then follow them to see who suffers a myocardial infarction.

Restriction allows people into a study only if they meet certain narrowly defined categories based on our suspected confounder. So, in the above example, we might restrict entry to only non-smokers.

Matching entails entering someone without the possible confounder for every person with the possible confounder, for example, for every non-smoker in the above study, we make sure there is a smoker.

9.5.2 Analytic Approaches to Confounding

Stratified Analysis is essentially the analytic approach we took above. We stratify the data into levels that are homogenous for the confounder under consideration and compare the crude (unadjusted) measure of association, to a measure of effect that takes this stratification into account. We will need a way to combine the effect measures from each individual level or stratum into an overall "adjusted" effect measure. (We will describe just such a statistic shortly.)

Multivariate analysis is perhaps the most popular statistical approach to confounding. The ease and power of computing makes procedures like logistic regression attractive options for all but the simplest data sets. We will visit this approach in our discussion of linear regression, when we will see that a β coefficient for an explanatory variable is interpreted as the effect of that variable holding constant all other variables in the model. We basically identify one or more of the variables other than the exposure variable as potential confounders. We then compare the "effect" of our exposure variable when the potential confounder is included or removed from the model. The key concept remains a comparison of crude with an adjusted measure of effect, i.e., compare model with the confounding variable to a model without the confounding variable

While we are here concentrating on the statistical aspects of confounding, it is crucial to not lose sight of first principles. Have a clear idea of what your (main) exposure variable is and what your outcome variable is. Make this decision early based on substance area knowledge. Causal thinking drives analysis, not the other way around

Now, back to our data about previous preterm labor and low birth weight.

9.6 Preterm Labor and Birth Weight Example 2

Say subject matter knowledge leads you to consider uterine irritability as a possible confounder of the prior preterm labor low-birth weight relationship. A review of the literature reveals that uterine irritability (a) may be causally related to low birth weight, (b) may be associated with previous preterm labor but (c) is unlikely to be a mediator or in the causal pathway between any previous preterm labor low birth weight relationship. So, it meets all the *a priori* conditions for a possible confounder. Our next step, is to evaluate through stratified contingency table analysis. Requesting a stratified analysis is as easy as adding the stratifying variable to your tables' statement:

```
proc freq data=birth nlevels;
  tables uterine_irr*prev_pretrm*low / measures;
  title 'odds ratio for association previous preterm and lbw';
  title2 'controlling for uterine irritibility';
run;
```

The following screen presents the odds ratio for the association between previous preterm labor and low birth weight for individuals without uterine irritability (Fig. 9.3).

The next screen presents the odds ratio for individuals with uterine irritability (Fig. 9.4)

We see that there is quite a noticeable difference in the OR for the association between previous preterm labor and low birth weight when uterine irritability is present or absent. What is going on here? Well, our first though should be that uterine irritability could indeed be a confounding variable. But, we should also consider the possibility of interaction or perhaps even mediation. We will need some additional statistical tests to help us assess the situation. But, as always, knowledge of our subject matter should be paramount.

9.7 Adjusted Odds Ratios

As noted in our discussion of stratified analysis for confounding, we need a statistic that validly combines the odds ratios from individual strata. The familiar (at least to

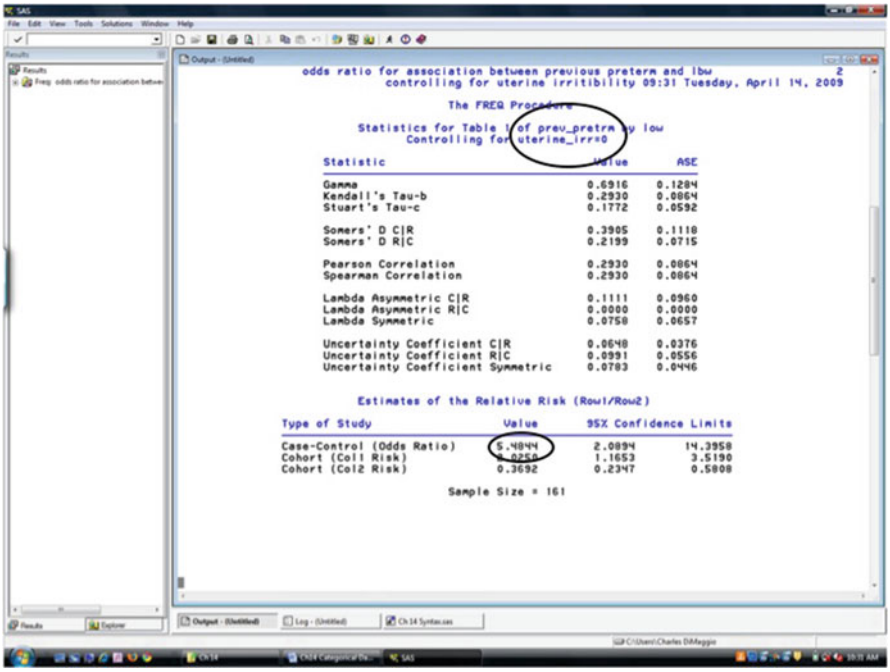


Fig. 9.3 Odds ratio for a stratum

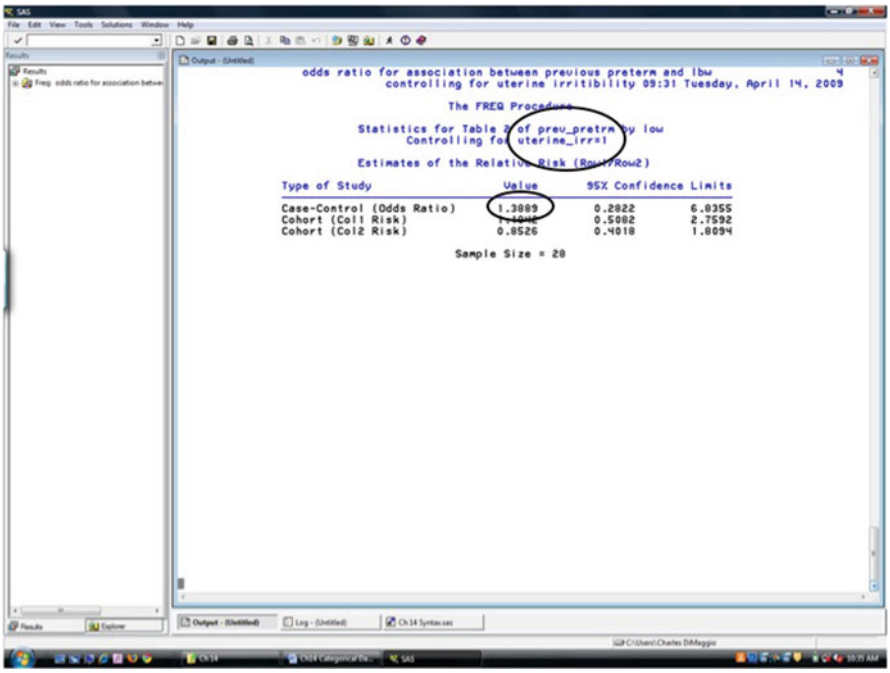


Fig. 9.4 Odds ratio for another stratum

Fig. 9.5 Types of CMH statistics in SAS

epidemiologists) Mantel–Haenszel odds ratio is just this statistic. SAS complicates matter somewhat by presenting two statistics that include the name "Mantel–Haenszel."

9.7.1 Cochran–Mantel–Haenszel Statistic

In SAS the Cochran–Mantel–Haenszel statistic (CMH) is a test for the *statistical significance* of an association. The Cochran-Mantel Haenszel statistic is a good test for associations. It doesn't require a huge sample size. (Any fairly large overall sample size will do). Be aware, though, that a large positive association at one stratum can affect a negative association at another stratum.

To make matters even more interesting, three are, in fact, 3 types of CMH in SAS. Type 1 is best for linear associations of ordinal by ordinal variables. Type 2 is best for raw mean scores of ordinal by nominal variables. Type 3 is for the general association of nominal by nominal variables (Fig. 9.5).

When interpreting your results, choose the CMH that gives the most power for your type of analysis, but note that while type 3 for general association has the least power, it is the one used most often. Again, the CMH is not the Mantel–Haenszel odds ratio with which most epidemiologists are familiar.

9.7.2 The Mantel–Haenszel Odds Ratio

The Mantel–Haenszel risk estimate combines the results of the stratum-specific comparisons into a single overall estimate. It is formed by a weighted average of the stratum effects where each weight is based on the precision of the effect and the size of the stratum which results in an adjusted measure of association or risk. SAS provides Mantel–Haenszel measures for both case-control and cohort data.

PROC FREQ returns 2 types of adjusted ORs for case-control data.[4] The Mantel–Haenszel (MH) estimator is a weighted average of stratum specific ORs that can

[4] You also get the same two types of estimates for cohort data.

handle zero frequencies. The logit-based estimator is a weighted average of log-odds ratios and zero frequencies are a computational problem. The procedure adds 0.5 to zero cells. SAS recommends that the MH estimator be used with small sample sizes.

It is important to note that the MH estimator is based on the assumption that ORs are constant across strata. This assumption is not met, for example, in the setting of interaction where effect estimates vary across strata of the third variable. It is important to consider and check for the homogenieity of the OR across strata.

The Breslow–Day statistic checks for the homogenieity of ORs across strata. It has a χ^2 distribution and requires an appropriately large sample size (recall, 80% cells >5, no cells <1). Note, though, that it can be misleading if the ORs vary across 1.

In SAS, the Breslow–Day statistic includes a so-called Tarones adjustment to correct for a perceived inefficiency of χ^2 when sample sizes are small. Requesting the Breslow–Day statistic also returns the CMH measures of statistical significance.

Simply including a "/ all" option[5] following your tables' statement returns the Mantel–Haenszel estimate. Including "bdt" returns Breslow–Day statistic and Tarones adjustment.[6]

We return to our birth data and the association between previous preterm birth and low birth weight for a demonstration of contingency table analysis.

```
proc freq data=birth nlevels;
   tables uterine_irr*prev_pretrm*low
         / all bdt; /* Tarone for small sample size */
   exact or comor;
/* exact preferred for small sample sizes
        comor gives CI for the exact OR*/
   title 'Contingency Table Analysis';
run;
```

While we get similar results to our previous runs, we will now also get exact results which may or may not be helpful. Lets jump to the adjusted results (Fig. 9.6)

Looking first at the CMH measures of statistical significance on the top part of the page, the type 1 statistic is most appropriate for our data, but we can just as easily use type 3.[7] The CMH statistic indicates that there is a statistically significant association between previous preterm labor and future low birth weight even when controlling for uterine irritability.

Turning our attention to the Mantel–Haenszel estimate for the common odds ratio across the two strata of uterine irritability, we see an association on the order

[5] You can also request / CMH rather than / all. If there is a 2×2 table, SAS will return the Mantel–Haenszel statistic in addition to the Cochrane–Mantel–Haenszel statistics.

[6] As you start to stratify by more than one variable, you may find that you will start having problems with small numbers. SAS has options to request exact tests for odds ratios. You request them with an "exact" statement on its own line.

[7] In fact, you wont go too far wrong just using type three routinely.

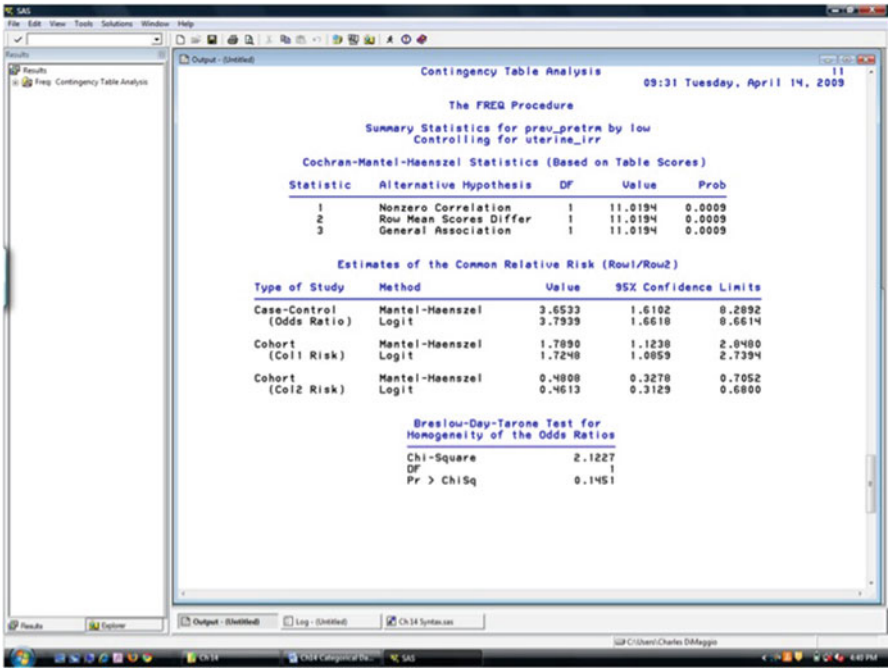


Fig. 9.6 Adjusted odds ratio

of about 3.7. This makes sense. The MH is a weighted average of the two strata. Since the association in the non-uterine irritability stratum was so large, it "pulled up" the low uterine irritability stratum.

Importantly, we note that the adjusted or MH OR of 3.7 is appreciably different from the crude OR estimate of 4.3.[8] Before we can conclude that uterine irritability is a confounder of the relationship between previous preterm labor and low birth weight and should be included in our model, we will need to check for evidence of interaction. Let's look a the Breslow–Day statistic for the homogenieity of the odds ratio which is found at the bottom of the screen. We see that the p-value associated with the Breslow–Day statistic is not statistically significant, indicating there is no evidence of statistically significant heterogeneity between strata and therefore little evidence of statistically significant interaction occurring.[9]

[8]There are no hard and fast rules for what constitutes an "appreciable" or "meaningful" difference between a crude and an adjusted estimate, nor is the comparison amenable to statistical testing. This is one of those areas (again) where training, experience and substance matter knowledge combine to guide our efforts. A 15% or 20% difference is an acceptable rule of thumb.

[9]Of course, the sample size was pretty small, and even with Tarone adjustment we may have to question these results. Also, see Chap. 11 for a discussion of an epidemiological approach to interaction. You would want to consider using the Darroch/Rothman approach.

TABLES

Epidemiological data are traditionally (and ubiquitously) laid out as 2 by 2 or Punnett tables:

	dis	nodis
exp	a	b
unexp	c	d

It is, actually, a simple enough task to enter data into SAS in terms of a 2×2 table:

```
data contingency;
   input exposure $ disease $ count ;
   cards;
   unexp nodis  15
   unexp dis    50
   exp    nodis 40
   exp    dis   20
   ;
```

But, SAS is very much geared toward the analysis of individual observations. SAS will read each of the cell counts as an individual observation. If you tried to run a PROC FREQ on these data, SAS would treat each of the cell counts as an individual observation and return a frequency of 1. We can turn SAS into an epidemiologist by adding a *weight* statement that treats the "count" variable the way we intend:

```
proc freq data=contingency;
   weight count;
   tables exposure*disease;
run;
```

The syntax for a stratified contingency table analysis would be:

```
/* stratified 2x2 table */
data stress;
   input confounder $ exposure $ outcome $ count @@;
   cards;
 conf unexp  dis 50 conf  unexp  nodis  10
 conf exp dis 100 conf  exp nodis  90
 noconf unexp  dis 60 noconf  unexp  nodis 140
 noconf exp dis  10 noconf  exp nodis  50
   ;
proc freq data=stress order=data;
   weight count;
   tables confounder*exposure*outcome /  all;
   run;
```

9.8 Summarizing Exploratory Contingency Table Analyses

As we've seen, the key to detecting confounding is comparing the crude to the adjusted odds ratios. For binary categorical data, even when you intend to create a multivariate logistic model, you should conduct frequency table analyses of all the variables in your model. An effective approach is to first use PROC FREQ to examine one-way and stratified cross tabulations of your outcome and exposure variables setting up a list comparing crude and adjusted measures for possible confounders and considering the possible role variables may play as interaction terms and mediators. Only then, should you move on to the multivariable analysis of categorical data.

Problems

9.1. Contingency Table Analysis
Using the safety data from the previous section compute chi-square tests of association and measures of the strength of association for type by safety and region by safety. Use the ALL option to request several tests and measures of association:

- What statistic could you use to measure the evidence of the association between type by safety?
- Is there statistical evidence of an association between type by safety?
- What statistic could you use to measure the strength of the association between type by safety?
- How would you interpret the statistic? What statistic could you use to measure the strength of the association between region by safety? How would you interpret the statistic?

9.2. Stratified Analysis
Perform a stratified data analysis on type by safety controlling for region. Use the ALL option to request several tests and measures of association:

- What statistic should you use to detect an association between type by safety controlling for region?
- How would you interpret the statistic?
- Why are the tables on Asia suppressed?
- Do you think this will cause problems in the model?

Perform a stratified data analysis on region by safety controlling for type. Request the Tarone's adjustment for the Breslow–Day statistic. Also use the EXACT statement and request exact confidence limits for the crude odds ratio and the adjusted odds ratio.

- Is there evidence that type is a confounder in the relationship between region by safety?
- Is there evidence that there is a type by region interaction?

Part III
Continuous Data and Regression

We address the analysis of continuous data after categorical outcomes only in deference to the usual practice of epidemiology. The consideration of continuous data and outcomes is, though, the traditional underpinning of biostatistical analysis. ANOVA introduces the concepts of regression analysis. Regression is perhaps the most frequently used statistical technique in biomedical research today.

Chapter 10
Cleaning and Assessing Continuous Data using MEANS, UNIVARIATE, and BOXPLOT

Abstract In this chapter we continue and expand on our brief introduction to continuous data from our consideration of PROC MEANS in Chap. 6. From an epidemiological perspective, the descriptive procedures in this chapter may be all that are needed to give us some summary statistics like means and medians, or simple graphical comparisons. They are often useful for data "cleaning" by providing tools to look for missing data, identifying outlier or erroneous values, and getting an overall sense of the data. These procedures, which in addition to PROC MEANS include PROC UNIVARIATE and PROC BOXPLOT, are also used to evaluate data for the assumptions for analyses such as ANOVA or linear regression. We will be talking (a lot) about statistical significance, variability, etc., and its easy to get caught up in the ideal of statistical significance as its own end. But, remember: clinical or epidemiological importance is what we're really interested in.

10.1 PROC MEANS (Redux)

We return to PROC MEANS as the most straightforward place to begin thinking about continuous, normally distributed independent data. Recall the basic form of PROC MEANS is:

```
PROC MEANS data =  ;
      VAR  variables;
RUN;
```

By default, PROC MEANS will return the number of non-missing observations, the mean, the standard deviation, as well as the minimum and maximum values. You can ask for things like the variance, quartiles, and the coefficient of variation. SAS help lists all the available options. It's also usually a good idea to print out the first 10 or so observations to get an idea of what you are working with.

C. DiMaggio, *SAS for Epidemiologists: Applications and Methods*,
DOI 10.1007/978-1-4614-4854-9_10,

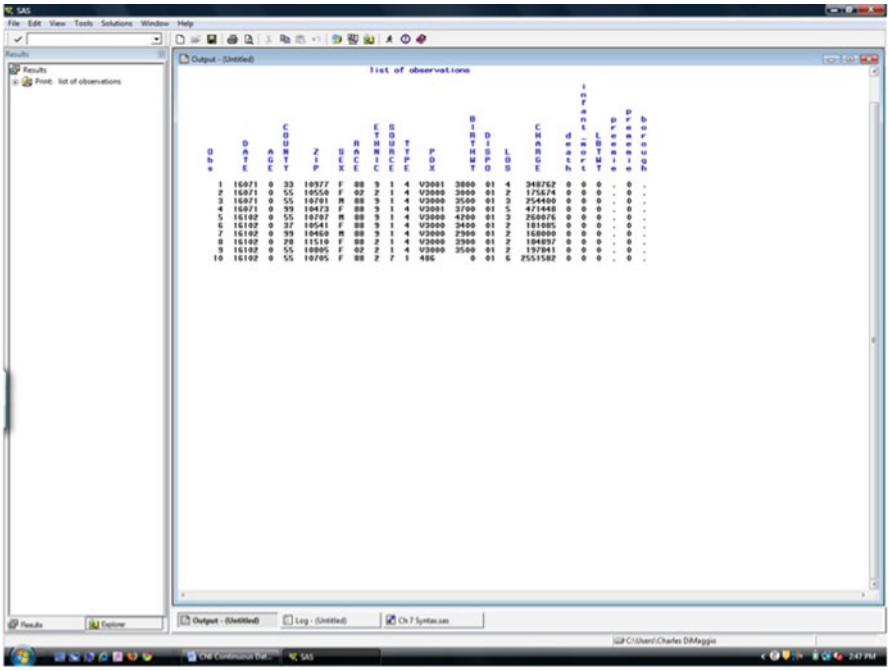


Fig. 10.1 PROC PRINT for a continuous variable

Let's quickly walk through the use of PROC MEANS for the initial evaluation of a continuous variable. We will work with a data set of children less than one year old discharged from New York City hospitals in our first step is to print out some observations (Fig. 10.1).

```
options nodate nonumber; /* some page options*/
proc print data=ch10.infants (obs=10);
     /* print only the first 10 observations*/
title ' list of observations';
run;
```

(What are two other ways you could get a sense of an unfamiliar data set?)[1]

Now let's run a PROC MEANS for the length-of-stay variable. If you want a statistic that is not part of the default set that the procedure returns, you must request the default statistics as well. Here, we want the variance, which is not part of the default set of statistics, so we also have to request the sample size, the mean etc. Anytime you want to add a statistic, you will have to go through this process of specifying the otherwise default statistics.

```
proc means data= ch10.infants
maxdec=4
```

[1] You could use PROC CONTENTS or the SAS Explorer.

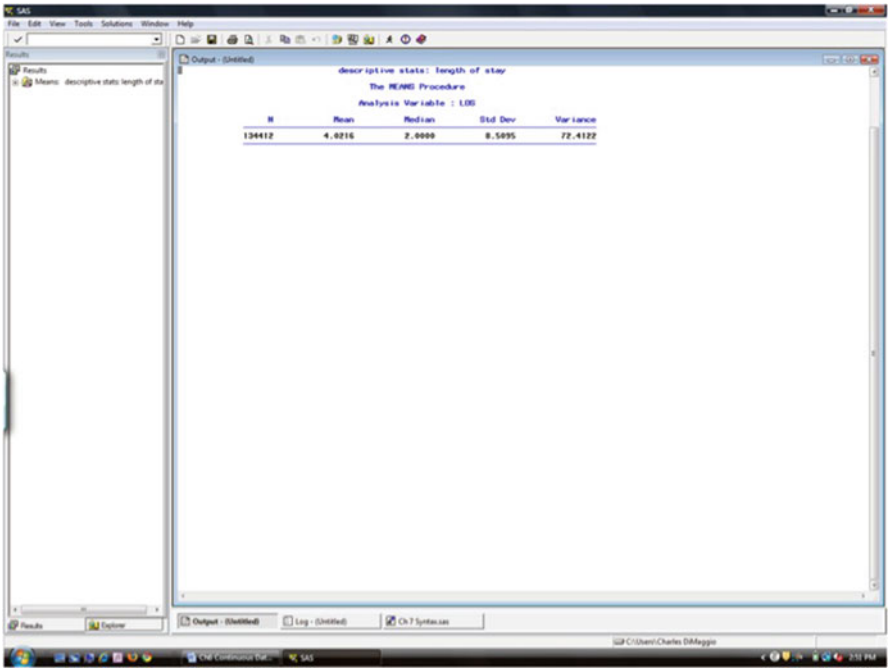


Fig. 10.2 PROC MEANS

```
n mean median std var;  /* proc statement ends with
semi-colon*/
var los;
title ' descriptive stats: length of stay';
run;
```

Even from this simple procedure, we can glean a fair amount of information about hospital length of stay for infants (Fig. 10.2). The mean length of stay was four days, but most children stayed two days. This tells us something about the shape of this data distribution (which we'll discuss momentarily) and whether it is possibly being influenced by one or more outlier observations. We see that the standard deviation is 8.5, which tells us about how tightly clustered (or not) the observations are around the central value of 4.

There are two very useful options in PROC MEANS with which you should be familiar. To request the standard error and the confidence limits of a mean, add STERR and CLM after the "data= statement" on the first line of the syntax. Remember, because you are requesting non-default statistics, you will also have to request the default statistics. The syntax looks like this:

```
proc means data= ch10.infants
        maxdec=4
        n mean median stderr clm var;
        var los;
        title ' descriptive stats: length of stay';
run;
```

N	Mean	Median	Std Error	Lower 95% CL for Mean	Upper 95% CL for Mean	Variance
134412	4.0216	2.0000	0.0232	3.9761	4.0671	72.4122

Fig. 10.3 Confidence interval for a mean

Running this syntax returns the following results (Fig. 10.3).

As expected, the mean and median for this run are precisely the same as our previous run. But now, rather than an estimate of about 8.5 for the standard deviation, we have an estimate of 0.02 for the standard error as well as upper and lower bounds for the confidence limits of the mean.

I hope you are learning to appreciate that the syntax for running procedures in SAS is actually not that difficult. The challenge is ensuring you are running them on data that makes sense, and just as importantly, understanding what the output means. To that end, before we move on to more sophisticated SAS procedures for continuous data, let's take a moment to review some basic statistics that arise from the calculations in PROC MEANS.

10.2 Review of Some Basic Statistics for Continuous Variables

There are some basic statistical concepts that are so fundamental to understanding data analysis that we sometimes run the risk of taking them for granted. One of these is, I think, how variability about some central estimate or a normally distributed set of numbers may be captured by sums of squares.

Begin by considering some sample of n numbers drawn from a population N, say $x_i = 1, 2, 3, 4, 5$. The mean of these numbers is simply their sum divided by n: $\bar{x} = \Sigma x_i / n$. Now say we're interested in knowing how closely clustered the values of x_i are around $\bar{x}$, perhaps because we're interested in how stable it is.

We could just subtract $\bar{x}$ from each value of x_i, and get the mean of these differences. You might take a moment to do this with some series of numbers using an Excel spreadsheet. Did you get zero? You always will, with any set of numbers. While this is kind of mind bending in its own right, it makes the prospect of getting a numeric estimate of the variation of a sample of numbers around its mean a bit more daunting, and leads to the idea of sums of squares.

Rather than just add up the differences, we first *square* the differences to ensure that all the values are positive, then we *sum* up these squared differences. We then

divide by one less than the total number of observations (see box below) to arrive at an estimate of the spread around the mean called the sample variance.[2]

$$s^2 = \Sigma \frac{(x - \bar{x})^2}{n - 1}.^3 \tag{10.1}$$

DEGREES OF FREEDOM

When calculating the sample variance, why do we divide by $n - 1$ rather than by n? Well, in practical terms, dividing by $n - 1$ brings the sample variance estimate closer to the population variance. This can be easily (though tediously) demonstrated by taking all possible 2-number samples of a series of 3 numbers. If we compare an average estimate of the variance when dividing by n to one when dividing by $n - 1$, we will find that the estimate based on $n - 1$ as the divisor is closer to the population estimate. In another practical example, we know we need at least 2 numbers to calculate a variance. If we had just one number to base our estimate on, the sum of squares would be zero, if we divided by n (n=1), the variance would be calculated to be 0/1 = 0. By using $n - 1$ instead, the formula would be 0/0 = undefined or meaningless, which is the correct answer when we are trying to estimate variance based on just one number.

On theoretical grounds, the choice of $n - 1$ is caught up in the concept of *degrees of freedom*. Degrees of freedom are the number of values in a calculation that are free to vary or (even more theoretically) the dimension of the vector space from which our calculation can be made. It is based on the very practical idea of how much information we need to determine the "location" of a scalar or vector. As an example, we know that the sum of the differences from the mean is constrained to be zero. If we know any $n - 1$ of the values, we will necessarily know the last one. They are constrained to a space of $n - 1$.

[2]In epidemiology we almost invariably consider our data to be samples. Even when we have everyone enumerated, we may consider it a sample of some *future* population. We will, therefore, in general, use sample nomenclature like s^2 for sample s for sample standard deviation as opposed to σ for a population standard deviation.

[3]In the increasingly unlikely event you find yourself having to calculate this by hand, there is a simpler calculating formula: $s^2 = \frac{(\Sigma x^2) - (\Sigma x)^2/n}{n-1}$.

We can take the square root of the sample variance to bring the estimate of spread back into the same metric as the original data with a statistic called the sample standard deviation[4]:

$$s = \sqrt{s^2}. \tag{10.2}$$

As I allude to in the previous footnote[5] standard deviations are not the only way to assess the spread or variability of a set of numbers. Another frequently used statistic, and one that SAS frequently reports, is the coefficient of variation (c.v.). The c.v. is the proportion or percentage of the mean represented by the standard deviation, i.e., c.v. $= s/\mu \cdot 100$. It gives you a quick sense of how large the sample standard deviation is compared to the mean.

A related, but distinctly different statistic from the standard deviation is the standard error. While standard deviation measures spread or *variability* of some set of numbers, the standard error is a measure of the *precision* of our estimate of some population parameter, here the mean.

Think of our estimate of the mean as being just one possibility drawn from some sample which is itself only one possible sample from the underlying population. Each possible sample from the population has it's own mean. Taken together, all these possible means, based on all those possible samples constitute the set of all possible means, of which ours is only one. How close or how far our humble sample mean is to the actual population mean is reflected in the standard error. We find that when we estimate the mean of a population from some reasonable sample, this sample mean will usually be the same (or very close) to the population mean itself. We also find that the precision of our estimate, how tightly aggregated the distribution of all possible estimates of the mean is around our chosen estimate, is related in a simple but powerful fashion to our sample standard deviation, which is itself related to sample size. The standard error is, essentially, the standard deviation of the sampling distribution of sample means and is equal to our sample standard deviation divided by the square root of the sample size:

$$s.e. = s/\sqrt{n}. \tag{10.3}$$

The standard error is an integral part of our calculations for p-values and confidence intervals. Notice, again, that the standard error is intimately connected to

[4]Standard deviation is so well accepted in statistics that it is, well, the standard approach to measuring spread around a mean. One might ask, though, why not just sum up the *absolute* differences and take the mean of that? Why not indeed. There are, in fact, other measures such as mean absolute deviation based on just that concept, which though not widely used, have their defenders (http://www.leeds.ac.uk/educol/documents/00003759.htm). Other measures of dispersion, such as mean absolute error (MAE), are preferred by practitioners of time series analysis and forecasting.

[5]You are studiously reading all the footnotes, aren't you?

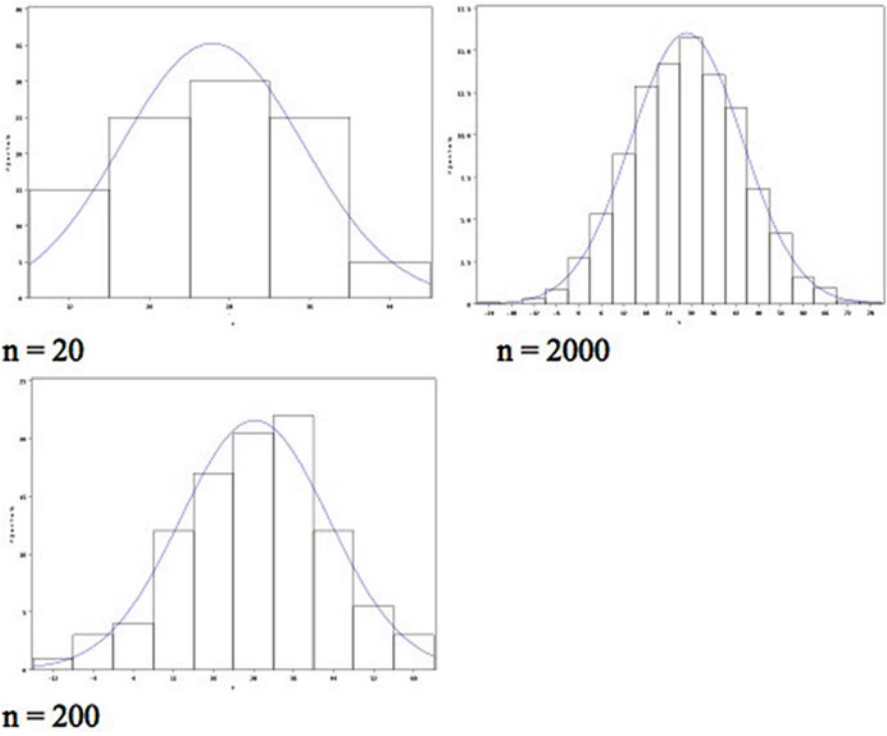


Fig. 10.4 Demonstration of the central limit theorem

the sample size, n. As sample size goes up, the standard error shrinks. Here are three histograms drawn from randomly generated numbers from a normal distribution. Notice how tight the distributions become as the sample size increases (Fig. 10.4).

STRENGTH IN NUMBERS.

The simple profundity of the relationship between standard error and sample size is not to be overlooked. We are saying that our estimate of the population parameter becomes more and more precise the larger our sample is. As information systems and data bases move toward the accumulation of ever larger numbers of data, standard errors will shrink. As we will see, we use standard errors to calculate things like confidence intervals. So beware. Large numbers mean smaller standard errors which translate into tighter confidence intervals and hence more instances of statistical significance. An appreciation for the difference between significance and importance was never so crucial.

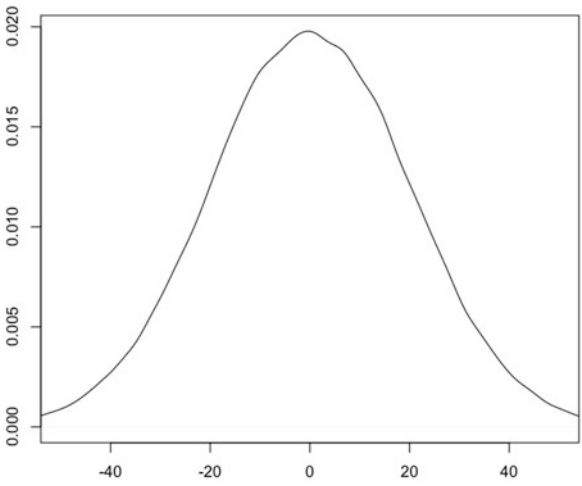


Fig. 10.5 Standard deviation of 20

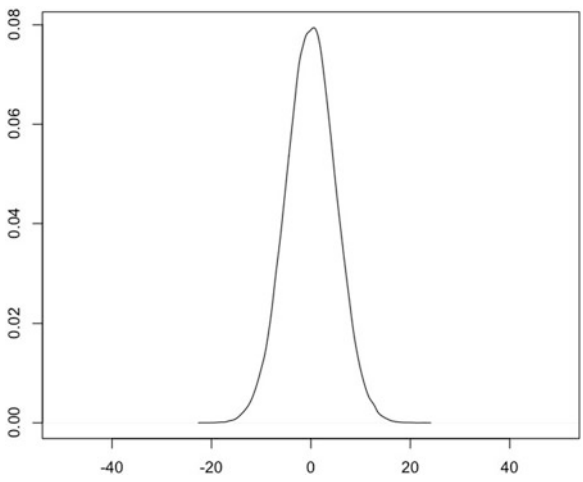


Fig. 10.6 Standard error of the mean $= \frac{20}{\sqrt{16}}$

Here's a graphical representation of the difference between standard deviation and standard error. The first figure is a density plot of a normally distributed population with a mean of zero and a standard deviation of 20 (Fig. 10.5).

This next figure demonstrates the standard error of the mean for this same population based on a sample of 16 (Fig. 10.6).

IMAGINE

The concept of all possible means based on all possible samples based on some mythical population of all possible people, may appear a bit fantastical but serves a purpose. I was once excoriated in a letter to the editor for using

(continued)

(continued)
the term standard deviation when I should have used standard error. Or was it the other way around. Anyway, besides demonstrating that some writers of letters to the editor truly do have too much time on their hands, I must admit to a certain statistical dyslexia regarding standard deviation and standard error, and to give the devil his due, there is an important distinction between the two that is well worth repeating. So again:

Standard deviation refers to the variability of a sample, population, or distribution itself. It is used when we want to know how widely dispersed or variable the sample, population, or distribution is.

Standard error (s.e.) refers to the precision of a parameter estimate. We apply it when we use a sample to estimate, say, a population mean and want to know how accurate or precise our estimate is.

And if the idea of all those "possible" samples sticks in your craw, perhaps Bayesian statistics is for you…

10.2.1 Confidence Intervals

You will have realized by now, that we have been considering an underlying normal distribution of values. We'll spend a little more time considering the implications and importance of the normal distribution in continuous data analysis in future chapters. For now, suffice it to say, that if your population is "normal" in the statistical sense, then about 68% of your data will lie within 1 standard deviation of your mean, 95% of your data lies within 2 standard deviations of your mean, and about 99% of your data lies within 3 standard deviations of your mean.

Another neat aspect about normally distributed variables is that the distribution of your sample means is also normal, which simplifies things like the calculation of confidence intervals. Confidence intervals are constructed as a point estimate plus or minus a critical value derived from a standard normal distribution (usually 1.96) times the standard error of your point estimate:

$$CI = \bar{x} \pm 1.96 \cdot s.e. \tag{10.4}$$

LIMITED CONFIDENCE

In so-called frequentist statistics we can't, despite the common-sense appeal of it, say that there is a 95% probability, or that we are 95% confident, that x lies within your CI. This is because x is considered fixed; it does not vary. What does vary is the CI itself. What we *are* saying is that if we

(continued)

(continued)
collected many samples (again, all those possible samples…) and calculated an estimate of x and its s.e. for each of those samples, 95% of the time the true x would be in our calculated CI.

If you'd like to just get the direct probability of x, again, maybe Bayesian analysis?

HOW NORMAL IS NORMAL?

Its amazing how many continuous processes are normally distributed. In fact, the Central Limit Theorem tells us that if we have enough data points the sample means of even non-normally distributed populations will be normally distributed. The larger the sample size, the closer to normality. A rule of thumb is about 30 data points. In social sciences, where data is noisier, we should have more data.

10.3 PROC UNIVARIATE

As we've seen, we can get a lot of information from the humble PROC MEANS. There is though a more useful procedure for continuous, normally distributed data that returns a much more informative standard set of additional descriptive statistics information and also gives you options for graphics like histograms (which are essentially like bar charts for continuous variables) and probability plots.

The syntax is straightforward. For our length-of-stay variable, it would be

```
proc univariate data= ch10.infants;
        var los; * if no var will analyze all variables;
        id date; * variable to use to identify the obs;
        histogram / normal;
        probplot;
        title ' univariate stats: length of stay';
run;
```

This syntax requests graphs, so you could precede it with a request to reset and prettify the graphic output. Something like this works fine:

```
goptions reset=all fontres=presentation ftext=swissb htext=1.5;
```

Notice a few things in comparison to our PROC MEANS runs of the length-of-stay variable. First, we get three pages of output, as opposed to a single line, and second, our graph window becomes active and displays two images. Let's explore some of this output.

First, as always, look at your log (Fig. 10.7).

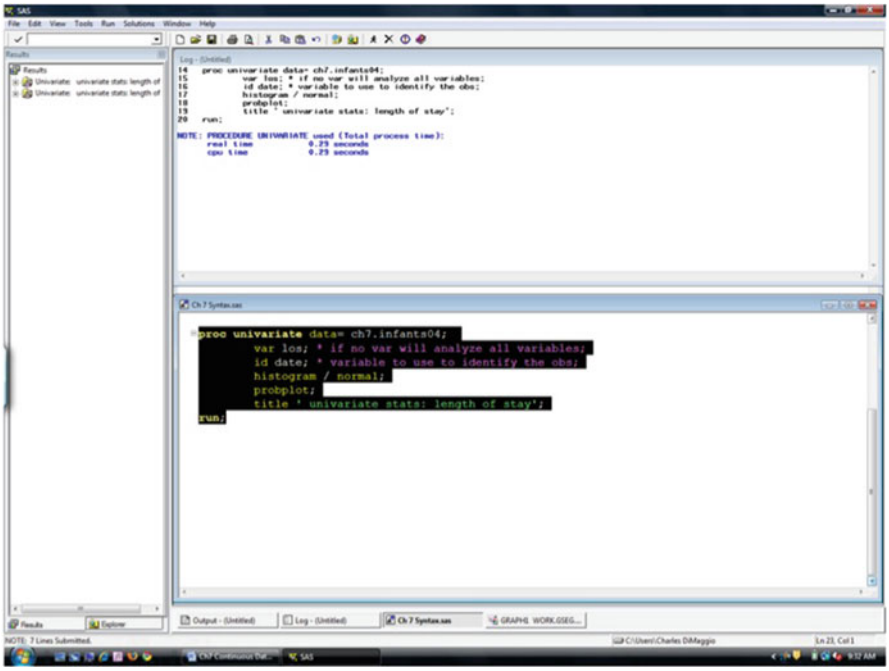


Fig. 10.7 Log output for PROC UNIVARIATE

Everything appears to be in order. Next let's look at our statistical results (Fig. 10.8).

Our first page of output returns some statistics familiar form PROC MEANS including mean, median, number of observations, standard deviation, and variance. The advantage here is that we dont have to request them; they are default statistics for the procedure. There are also a couple of new things here. We get, by default, the coefficient of variation. We also get a skewness and kurtosis statistic.

Skewness is the tendency for the distribution of values to be more "spread out" or "longtailed" on one side than the other. If a distribution is normally distributed, we would expect (and want) the skewness statistic to be at or close to zero. We can usually tolerate anything up to 1 or down to -1.

When the distribution is more spread out on the left side (your left, i.e., as you look at the page), it is called (logically enough) *left skewed* and the skewness statistic is *negative*. We will also see that when left skewed, the *mean is less than the median*. This is a useful hint if you don't have a skewness statistic with which to work.

A *right-skewed* distribution is heavier tailed or "spread out" on the right side. The skewness statistic is *positive,* and the *mean is greater than the median*. That appears to be the case here.

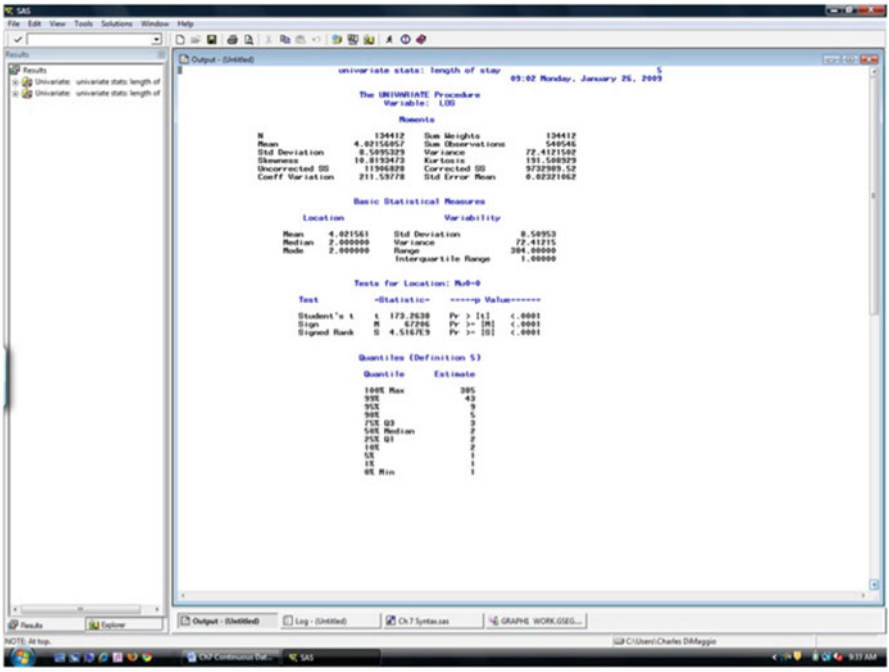


Fig. 10.8 PROC UNIVARIATE statistical output

Kurtosis refers to how peaked a distribution is.[6] Again, we would want the statistic to be close to zero. A *positive* kurtosis statistic indicates a *high-peaked*, heavy-tailed distribution. A *negative* kurtosis statistic indicates a *smaller-peaked,* light-tailed distribution.

This page of results also presents quantiles for the data set. *Quantiles* are a value below which some proportion of the distribution lies. For example, 3 values (the quartiles) divide a distribution into 4 equal parts. The *interquartile range* is the difference between the first and the third quartiles and can give you an indication of how tightly clustered or spread out data is. Quantiles are returned as a default part of PROC UNIVARIATE output.

[6]There are some neat names for kurtosis that might come in handy if you are trying to impress a statistician. A high-peaked, heavy-tailed curve is leptokurtotic. A small-peaked, light-tailed curve is platykurtic. When the kurtosis statistic is close to zero, the curve is mesokurtotic.

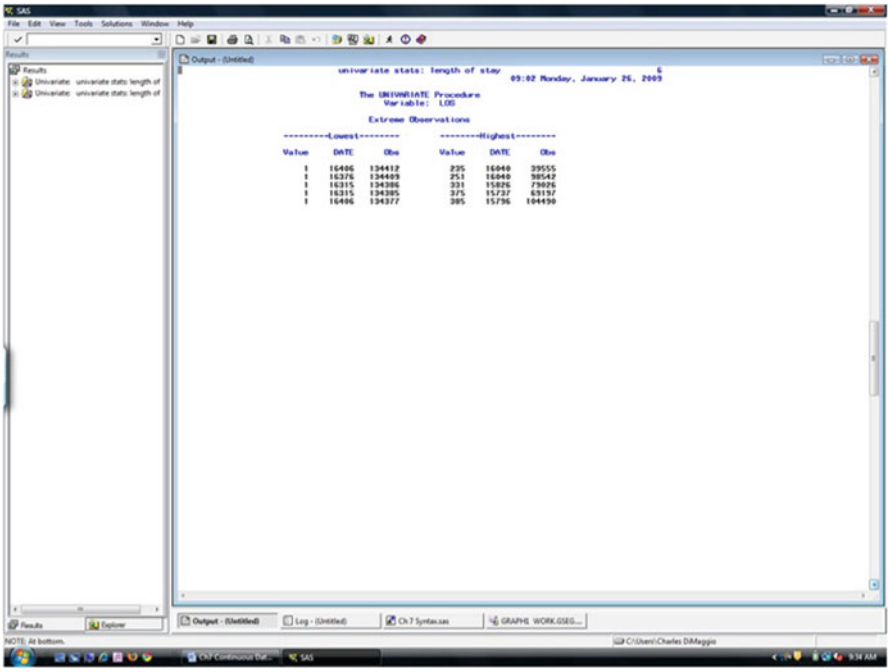


Fig. 10.9 PROC UNIVARIATE extreme values

STEMS AND LEAVES

Back in the not so very distant past, statistics were calculated by hand. John Tukey (who we'll meet again) developed a neat way to summarize data by grouping numbers by their digits. So 36 would have a stem of 3 and a leaf of six. A stem and leaf plot for a set of numbers might look like this:

2| 01189
3| 1255568999
4| 23

It not only lists all the observations but gives an indication of the distributions shape. You can request a stem and leaf PLOT in PROC UNIVARIATE:

```
PROC UNIVARIATE data=your.data  PLOT;
var num_var;
run;
```

On this next screen, we are given the five highest and five lowest observations for length of stay (Fig. 10.9). Some newborns only stayed one day. There was at least one, though, who was an inpatient for more than a year. Depending on what our outcome of interest is, this may be an outlier or perhaps an erroneous entry. We will want to investigate that more closely. Notice we asked SAS to identify

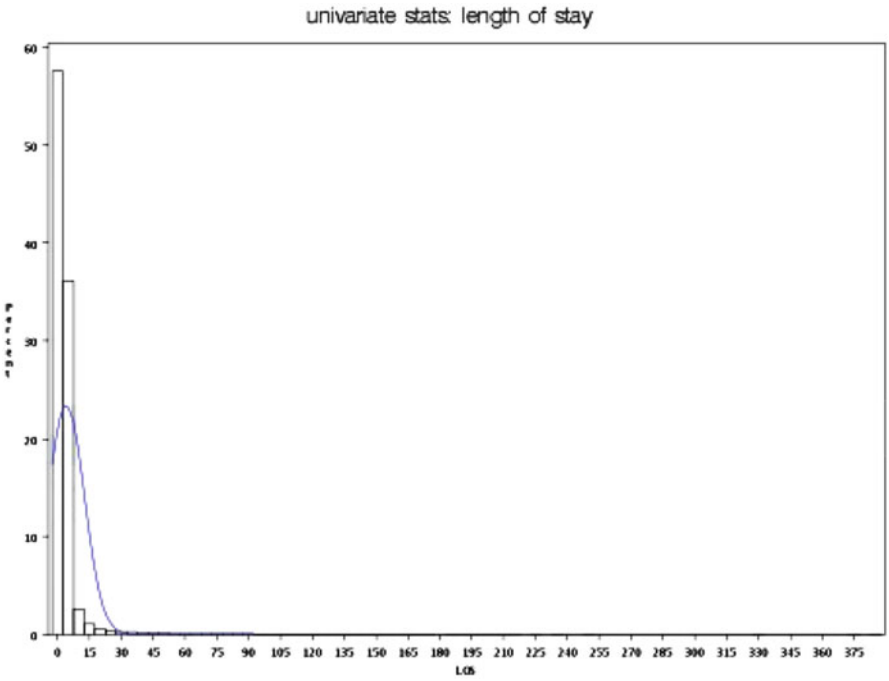


Fig. 10.10 PROC UNIVARIATE histogram

the observations by the date variable. In this example it is purely for demonstrative purposes. I just wanted you to see how the ID option in PROC UNIVARIATE works. It could, though, prove to be more useful in tying an outlier to some more helpful variable than just observation number.

There is an additional screen of output related to how well the distribution approaches a normal distribution which we will not consider. We derive, I think, enough information about normality from the above statistics and the following graphs.

Our histogram statement with the request for an overlying normal plot on the 4th line of the syntax returns the following graph. It is quite clearly not normally distributed (in fact, it looks Poisson distributed). Consider the form in light of our skewness and kurtosis statistics (Fig. 10.10).

Our next graph is the result of our request for a normal probability plot (PROBPLOT) (Fig. 10.11). A normal probability plot is a tool to assess if a set of data is normally distributed. You arrange your data in ascending order and plot it on the *y*-axis against the *z*-values for the data points on the *x*-axis. You expect, if the data is normally distributed, a straight diagonal line ascending from the lower left-hand corner of the plot to the upper right-hand corner. If the data is not normally distributed, you will see a curve of some kind. A single bend, as in the above example, indicates skewness. More than one curve indicates possibly a bimodal distribution.

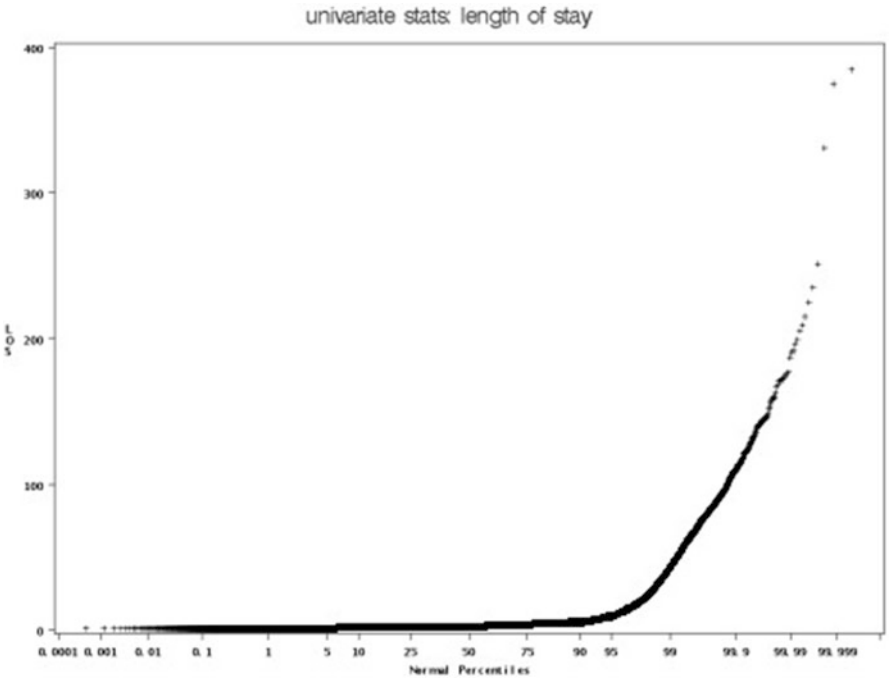


Fig. 10.11 PROC UNIVARIATE Normal Plot

We could also have included as part of the probplot statement the "/ normal'option" to request that SAS superimpose a reference line as we did with the histogram.

Similar to how you can request a confidence interval about the mean of some value in PROC MEANS, you can request a confidence interval in PROC UNIVARIATE with the option CIBASIC on the data line. You can set the specific type I error level with an alpha = option:

```
proc univariate data=your.data cibasic alpha=0.01;
```

10.4 PROC BOXPLOT

We will consider one additional tool in our continuous data assessment and cleaning armamentarium. PROC UNIVARIATE will return a crude box and whisker plot of your data as part of the same PLOT option that returns a stem and leaf plot (see above). A box-and-whisker plot returns the mean, median, quartiles, and minimum and maximum values.

PROC BOXPLOT will return a much neater looking graphic of this convenient and informative 5-number summary. It is useful for comparing the continuous variable we are interested in across groups. The syntax looks like this:

```
PROC BOXPLOT Data = ;
           PLOT continuousVar * group var  / options;
RUN;
```

The "continuousVar" above is the continuous numeric variable you want the box plot on. You can request box plots on more than one variable, but you need to enclose them in parentheses. The group var above is the categorical variable across which you want to examine the box plots. It needs to be numeric and categorical.

You must sort the data set by your grouping variable before running PROC BOXPLOT.

Note that SAS requires a grouping variable as part of PROC BOXPLOT. If you were interested, say, in looking at a boxplot for the entire data set, you would have to create a dummy variable to group the continuous variable.

To create a dummy variable, you use a DATA step to create a data set that includes the dummy variable:

```
DATA new_data_set;
  SET existing_data_set;
  dummy = 1;
/* create dummy variable to do boxplot on just one group */
run;
```

You can include options after the PLOT statement to improve the appearance of your graph. The option "cboxes" specifies the color of the boxplot, the option "boxstyle" specifies type of boxplot. The following syntax would return a pleasing enough graphic:

```
plot continous_var*group_var / boxstyle schematic
  cboxes=black;
```

As with any SAS graphic, the trick is to find the options and settings that serve you well and then stick with them. I find the following syntax nice:

```
symbol color = salmon;
   title Boxplot';
   proc boxplot data=work;
      plot cont_var*cat_var / cframe   = vligb
                              cboxes   = dagr
                              cboxfill = ywh;
   run;
```

To get a quick look at PROC BOXPLOT in action, lets return to our "preemie" data set. Say were interested in how length of stay varies across different discharge dispositions. As before, LOS is our continuous outcome variable. The variable DISPO would be our numeric grouping variable. As noted, we first have to sort by our grouping variable:

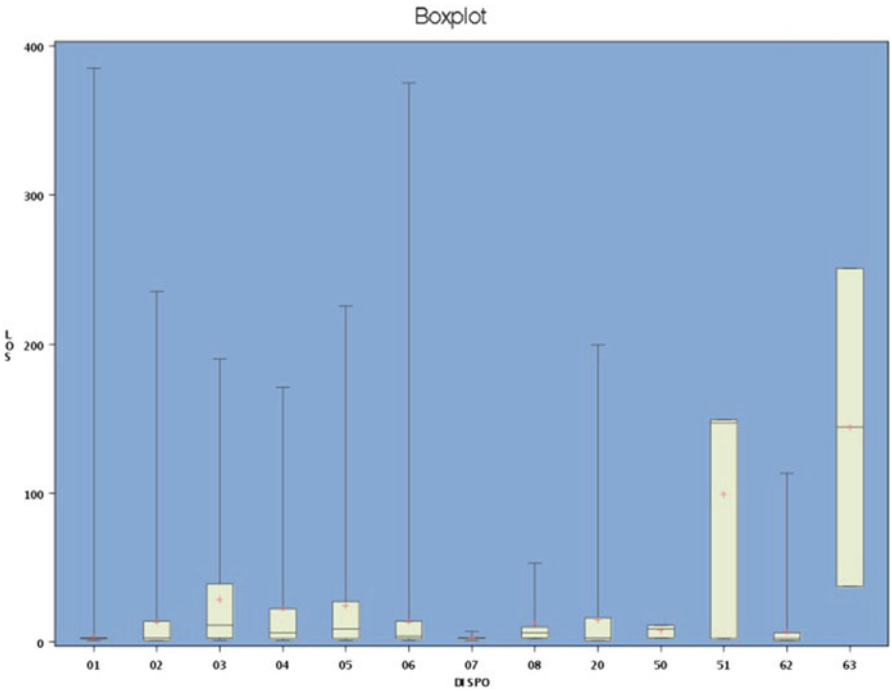


Fig. 10.12 PROC BOXPLOT

```
proc sort data=ch10.infants;
by dispo;
run;
```

We would, of course, check our log to make sure the process went smoothly. We then run the following syntax:

```
symbol color = salmon;
   proc boxplot data=ch10.infants;
      plot los*dispo / cframe  = vligb
                       cboxes   = dagr
                       cboxfill = ywh;
      title 'Boxplot';
run;
```

which returns the following graphic (Fig. 10.12).

There is really quite a bit of information here. First, each box plot gives an idea of the normality and symmetry of the continuous response variable. Think of flipping the box plot clockwise onto its right. If right or positively skewed you will see the upper whisker longer than the lower whisker. You will also see the median, represented by the horizontal black line, below the mean, represented by the red plus sign. You will also get a quick sense of outliers. The maximum length of the whiskers is about 1/2 times the interquartile range (from the top of the box to

the bottom of the box). Anything beyond that is represented as outliers. It is clear from the above example that we would have to seriously reassess any assumption of normality of length of stay across discharge dispositions.

10.5 In Summary

The three procedures presented in this chapter, MEANS, UNIVARIATE, and BOXPLOT, are useful in the initial analysis of continuous variables. They not only provide useful descriptive information; they allow us to assess the basic form of our data so that we can choose the most appropriate model or perhaps transform our data to fit the assumptions of the model we are planning to use.

Problems

10.1. PROC MEANS
Use the SPARCS text file to create a temporary (work) SAS file called infants by reading in the following variables: date, age, county, zip code, sex, race, ethnicity, source of admission, type of admission, primary diagnosis, birth weight, disposition, length of stay, and charge.

- Use "disposition = 20" to create a death indicator. Use age less than 1 and "disposition=20" to create an infant mortality indicator.
- Use type of "admission = 4" and birthweight less than 2,500 grams to create a low birth weight indicator.
- Use type of "admission = 4" and source of "admission = 2" to create a premature birth indicator.
- What is the mean birthweight for neonates? Does anything strike you as unusual about this result? (Hint: What is the normal birth weight?)
- How many observations were used to calculate this result? Where do you think you got this result? (Hint: What patients are included in this data set?)
- Rerun your analysis of birthweight on this data set. Request the mean, the median, the standard deviation, the variance, and the confidence limit for the mean. What can you tell from these results? (Hint: What was the mode?).
- Use a sub-setting IF statement as part of a DATA step to create a permanent file restricted to children less than 1 year old. How many children less than 1 year old were discharged?
- Run a PROC MEANS statement on this file requesting the mean, the median, the standard deviation, the variance, and the confidence limit for the mean. What are your results, and how do they differ from your first run?

10.2. PROC UNIVARIATE

Using the infant file you created above, run a PROC UNIVARIATE on the birthweight variable. Begin your syntax by resetting your graph parameters and specifying presentation level font resolution, swissb as the text and a text height of 1.5.

Request a stem and leaf plot, a histogram, and a probability plot with a normal overlay using an estimated mean and sigma (choose an appropriate color and width). As part of the univariate run, test the null hypothesis that the mean birth weight was 2,500 grams.

- Does the data appear normally distributed? What leads you to your conclusion?
- What is the skewness statistic, and what does it indicate?
- In what way do the mean and median reflect this?
- What is the kurtosis statistic, and what does it indicate?
- What can you learn from the extreme values that might provide information about lack of normality?
- Rerun your analyses excluding zero values with a sub-setting WHERE statement. How does this affect your results?

10.3. PROC BOXPLOT

Use PROC BOXPLOT to examine whether birthweight varies by county. Start with your infants04 data set. Use a DATA step to create a data set where NYC county is re-coded as a new variable called borough as follows: county = “58,” borough = 1; county =“59,” borough = 2 county =“60,” borough = 3; county = “61”, borough = 4; and county = “62,” borough = 5. Make sure to code any other value of county as missing.

Create boxplots of birth weight by borough. What, if anything, do you conclude based on these plots?

Chapter 11
ANOVA

Abstract In this chapter, we continue our consideration of continuous outcome variables. We now address the use of analysis of variance (ANOVA) in the situation where we have one or more categorical predictor variables. We'll review the concepts underlying ANOVA and describe how to apply it in SAS with PROC GLM. Well introduce the concept of error terms. We wi'll also take the opportunity to go over the concept of interaction in epidemiological studies

11.1 Review of ANOVA

To motivate our discussion[1] of ANOVA, consider the situation where you have two or more groups of individuals and you are interested in comparing them on some continuous outcome. Perhaps you want to compare the systolic blood pressures of attendees of three different clinics. As you most likely already know, it is not appropriate to do multiple tests of significance. We'll review why this is so, but suffice it to say that in this setting ANOVA is the more appropriate approach.

ANOVA is a statistical model appropriate when you have a continuous response variable and one or more categorical predictor variables. The ANOVA approach makes use of F-tests. The F distribution is, in fact, based on the χ^2 distribution, which is the sum of the squared differences of observed minus expected values divided by the expected value:

$$\chi^2 = \frac{\Sigma(O-E)^2}{E}. \tag{11.1}$$

[1] And motivation it may require. I do not use ANOVA very frequently in my epidemiological work. My usual "go to" procedure for continuous outcomes is linear regression. But, as we will see, ANOVA and linear regression are really two sides of the same coin. And since ANOVA emphasizes the effects of categorical variables, with which epidemiologists are so frequently concerned, it is a useful and informative starting point for regression-based techniques.

C. DiMaggio, *SAS for Epidemiologists: Applications and Methods*,
DOI 10.1007/978-1-4614-4854-9_11,

The F distribution is defined as the ratio of two independent χ^2 's divided by their respective degrees of freedom.

In the setting of ANOVA, we use the F distribution to test the hypothesis that *two variances are equal*. We divide the result for one variance by that of the other and look up the probability of the resulting statistic in a table of F statistics.[2] We obtain the variances we use in the division by partitioning and comparing two different components that make up the total variance in a data set.

Say, for example, that we have blood pressure readings on all the clients of three clinics. We calculate the total variance (or *total sum of squares*[3] in the data set by summing up the squared values of the differences between each individual blood pressure value and the overall mean blood pressure value for the entire data set. We then partition this total variance into two parts. First, we calculate a variance term for the group means by summing the squared differences between the group means and the overall mean for the entire data set. This is referred to as the *sum of squares between* groups or sometimes the *model sum of squares*. Second, we calculate the variance for each clinic or group of patients by summing up the squared value for the differences between each individual in a group or clinic and the mean blood pressure value for that group or clinic. This is referred to as the *sum of squares within* groups, but also goes by the name of *residual* or *error* sum of squares. These three values are clearly and intuitively related: $SS_{\text{Total}} = SS_{\text{Between}} + SS_{\text{Within}}$, where, for individual observations i in groups j,

$$SS_{\text{Total}} = \Sigma(x_i - \mu_{ij})^2 \tag{11.2}$$

$$SS_{\text{Between}} = \Sigma(\mu_j - \mu_{ij})^2 \tag{11.3}$$

$$SS_{\text{Within}} = \Sigma(x_{ij} - \mu_j)^2. \tag{11.4}$$

As in most traditional statistical testing, we begin from the view that there is no difference among the groups, for example, our null hypothesis (H_0) might be that the mean blood pressure readings are the same for all our clinic groups. If this is true, then the ratio of the measure of the variance of the group means (SS_{Between}) to the measure of the common variance for all the groups (SS_{Within}) should be 1. The more it deviates from 1, the more likely that it is that the group means are not, in fact, the same, and the more significant will be the result of an F-test that compares the between group variance to the within group variance.

To conduct the F-test, we divide the sums of squares by their appropriate degrees of freedom. The approach is laid out in the familiar[4] ANOVA table.

If the F-test in the ANOVA is statistically significant, we can then look for which particular means differ from the others by using multiple comparison procedures

[2] Or at least we used to. Nowadays, a machine looks it up for us.

[3] Another example of the use of that most fundamental of statistical concepts, the sum of squares.

[4] To some

Source of variation	Sum squares	d.f.	Mean square	Variance ratio
Between groups= SSB	$\Sigma(\mu_j - \mu_{ij})^2$	$n-1$	SSB / d.f.	SSB / SSW
Within groups = SSW	$\Sigma(x_i - \mu_j)^2$	$n(m-1)$	SSW / d.f	
Total	SSB + SSW	$n(m-1)$		

such as the Bonferroni method, the Scheffe method, Tukey's Honestly Significant Difference, or the Newman–Keuls procedure.

11.1.1 Assumptions for ANOVA

There are important assumptions underlying the use of a procedure like ANOVA. Specifically, we are assuming that (1) our outcome variable is indeed continuous, (2) that our observations are independent, for example, that taking a blood pressure measurement on an individual will not somehow affect or is related to taking a blood pressure measurement on another individual, (3) that the outcome is normally distributed, and (4) that the variance of the outcome is equal across groups.

ANOTHER VIEW

In preparation for our future discussion of regression analysis and regression diagnostics, it can be helpful and useful to take a somewhat different view of ANOVA. We can, without losing any information, view ANOVA as a model, where:

Outcome = overall average + differential effect from categorical variable + error.

Note that *the error term is the only random variable on the right side of the equation* This error term is (if our assumptions about the outcome variable are correct) therefore random, normally distributed, with equal variances across groups. We will find that it is often easier to test our assumptions of normality etc on the *error* term than on our continuous outcome variable. This is particularly the case when we get to multiple linear regression.

A DIFFERENT VIEW.

In more general terms, an error or residual value, can be thought of as the difference between an observed value and that predicted by the model. They are denoted by r or $\hat{e}$. Again, they should be random, normally distributed

(continued)

(continued)
with $\mu = 0$. Regression diagnostics consist in large measure of feeding these residual values into PROC UNIVARIATE and assessing their normality, etc.

Note that the Within Group Sum of Squares, i.e. $\Sigma(x_i - \mu_j)^2$, is also referred to as the Residual or Error Sum of Squares, because it is essentially defined by these errors.

11.2 Testing Assumptions with MEANS, UNIVARIATE, and BOXPLOT

A continuous response variable with a single categorical predictor variable is referred to as a one-way ANOVA. Basically, you are comparing some mean across two or more groups.

We are obligated to test our assumptions of normality, independence, etc., before proceeding to our main event of the ANOVA. SAS makes it fairly painless to do that through the use of descriptive procedures like PROC MEANS, UNIVARIATE, and BOXPLOT. Testing your assumptions has the additional advantage of allowing you to "get to know" your data and present overall descriptive measures in which your audience will very likely be interested.

While PROC MEANS is a simple enough place to begin, we usually take advantage of the convenience of PROC UNIVARIATE to first analyze our continuous outcome variable.

```
proc univariate data = your data;
     class cat_var;
     histogram cont_var / normal;
     var cont_var;
     probplot cont_var / normal  (mu=est sigma=est color=blue
       w=1);
     title univariate analysis by categorical variable;
run;
```

We use the CLASS statement in PROC UNIVARIATE to categorize the data according to the group variable in which we are interested. We are, therefore, assessing normality for each group. We will look at all the statistics that UNIVARIATE so conveniently provides, like comparing means and medians, kurtosis, and skewness. Note that we are also requesting a histogram with a normal overlay and a probability plot to assess normality.

Our next step will be to use PROC BOXPLOT to get a better visual representation and convenient 5-number summary for the continuous outcome data for each group. Remember, you must first sort your data by the classification or grouping variable.

```
proc sort data=your.data out=sorted_data;
/* need to sort data by your category variable*/
    by cat_var
run;

proc boxplot data=sorted_data;
       plot cont_var*cat_var  / cboxes=black
       boxstyle=schematic;
run;
```

11.3 ANOVA with PROC GLM

OK. There is a SAS procedure called PROC ANOVA, but I do not use it to conduct ANOVA. It has a somewhat limited number of associated options and really only works for the most straightforward situation of a so-called balanced ANOVA design, i.e. when you have the same number observations each group. I find this a bit limiting.

It is more convenient to use a procedure called PROC GLM that will handle any ANOVA design (as well as ANCOVA, or the analysis of covariance, and linear regression). Also, PROC GLM has a nice set of options and tools that allow us to easily create a data set of residuals or error terms for normality testing with UNIVARIATE.[5]

GENERALLY SPEAKING

GLM refers to *general linear model*, which includes ANOVA and all its linear cousins like linear regression, ANCOVA (analysis of covariance), MANOVA (multivariate analysis of variance), and MANCOVA (multivariate analysis of covariance). All these procedures are related in that they allow prediction of a continuous variable that is linearly related to predictor variables.

The general linear model can be further generalized to the appropriately but slightly confusingly named *generalized* linear model, which allows analysis of problems where linear assumptions are not appropriate, for example, non-continuous outcome variables, nonlinear associations with predictor variables, non-normally distributed data, and nonconstant variance across observations. Generalized linear models are characterized by a *link function* that characterizes how the outcome variable is related to the predictor variable. So in a traditional (or general) linear model with a continuous outcome variable with a normal distribution, the link function is a so-called identity function. When

(continued)

[5]This is not always the case. For a number of other SAS PROCS, you have to create the residuals by hand before feeding them into UNIVARIATE.

(continued)
the outcome is a proportion with a binomial distribution, the link function is a *logit* or $Pr[p/(1-p)]$, and the procedure is referred to as logistic regression. If the outcome variable is a Poisson distributed count variable, the link function is the natural log of the mean ($ln(\mu)$). In SAS, such generalized models can be run under PROC GENMOD.

Conducting a PROC GLM run for ANOVA basically proceeds in three steps:

1. Run the model with PROC GLM
2. Plot the residuals with PROC GPLOT
3. Get statistics on the residuals with PROC UNIVARIATE[6]

To run the model:

1. Invoke the procedure and specify the data set: *PROC GLM data = .*
2. Specify your grouping variable: *CLASS categorical variable.*
3. Specify your model: *MODEL continuous outcome variable = categorical variable.*
4. Compute means for each group: *MEANS categorical variable.*
5. Output your residuals as a data set: *OUTPUT out = error dataset.*
6. Run your model: *RUN.*
7. Quit GLM: *QUIT.*

We will spend some time assessing whether the data meet the underlying assumptions for ANOVA. Recall that we generally look at the error terms as a reflection of the overall model fit.

We will use PROC GPLOT to plot residuals against predicted values to see if there are any patterns that violate the assumption that they are randomly grouped around zero. On the plot, you should get a spike of data for each group (e.g., if only two groups, only two spikes of data). Look to see if the data points are evenly spaced on either side of zero.

After plotting the residuals, run PROC UNIVARIATE on them. Look at the normality statistics. The mean, median, and mode should all be close to zero, as should the kurtosis and skewness statistics. Request a histogram and examine it to see if the data appear normally distributed. Request a normal probability plot and examine it to see if it is a straight line.

[6]You may notice that we are running the model before testing our assumptions. This is not, strictly speaking, the best approach, but we need to run the model to get the residuals. So it is basically unavoidable. You will, of course, be tempted to peak at your results. Shame on you.

It is only after this important preliminary work that you will turn your attention to the results of the model itself.

The following syntax can be used as a template to run PROC GLM for ANOVA:

```
options ls=75 ps=45;  /* page and line spacing options*/
proc glm data = your.data;
   class cat_var; /* categorical variable to
                  group the outcome variable*/
   model cont_var=cat_var; /*specify the model i.e. response
                           variable = predictor(s) */
   means cat_var / hovtest; /* computes means of dependent
                                  variable for group*/

/* hovtest is Levene's test for homogeneity (equality) of
  variances
 (one of the assumptions of ANOVA is homoscedasticity)
  null hypothesis is variances are equal, do not want to reject
    null,
  want a large p value (look under 'Pr > F') on output*/

  lsmeans cat_var / pdiff=all adjust=tukey; /* pdiff=all
                           requests all pairwaise p values */

   output out=check r=resid   p=pred;  /* creating an output
                              data set called 'check', 'r '
                                is a
                              keyword SAS recognizes as
                              residuals, p is recognized as
                              predictors */
title 'testing for equality of means with GLM';
run;
quit; /* have to quit out of glm, or will keep running */

/* run gplot on the 'check' dataset created above*/

proc gplot data=check;
   plot resid*pred / haxis=axis1 vaxis=axis2 vref=0; /* can
             leave this out if you're OK with defaults*/
   axis1 w=2 major=(w=2) minor=none offset=(10pct);
   axis2 w=2 major=(w=2) minor=none;
   title 'plot residuals vs predictors for cereal';
run;
quit;

/* run proc univariate to get histogram,
      normal plot, kurtosis and skewness on residuals*/
proc univariate data = check normal;
  var resid;
  histogram / normal;
  probplot / mu=est sigma=est color=blue w=1;
  title;
run;
```

11.3.1 GLM ANOVA Output

If you are reasonably sure your data meets the assumptions of ANOVA, you may then turn to the actual GLM output. The first few lines of the GLM output consist of administrative information such as the variable chosen as the class variable, how many levels are in this class variable, the number of observations and how many of those observations, were used in the analysis.[7]

The next section of output contains all the information you need to test the *equality of the means* of your continuous outcome variable *across the class or group variable*. Remember, you are interested at this point in determining whether the group or class variable is an important predictor or determinant of the mean for the outcome.

You will find the classic ANOVA table where the first column indexes the sources of variability (total, model or between, error or within),[8] the next column presents the sums of squares associated with the sources of variability, and the third column represents their respective mean squares which are simply the sums of squares divided by the appropriate degrees of freedom.

Your evaluation will be led inexorably to the F statistic which is based on the model or between mean square divided by the error or within mean square.

$$\text{F} = \text{Model (Between) MS / Error (Within) MS}$$

Recall your null hypothesis is that *all the group means are equal*, i.e., group status has no effect or relationship to the mean of your continuous outcome variable. A large F is evidence that between group variability is greater than within group variability and indicates that the model or group status is important. SAS gives you the p-value for the F statistic, with a smaller value indicating that the observed results are unlikely to have occurred due to chance.

You should next look at the R^2 statistic[9] which is a measure of the proportion of total variability accounted for by the model or group status. It is calculated quite intuitively as the model or group sum of square divided by the total sum of squares.

$$\text{R}^2 = \text{Model (Between) SS / Total SS}$$

It is a measure of how well the model fits, explains, or accounts for the variability in the data. The more of the total variability accounted for by the model (or group status), the better the fit. So, values closer to 1 are more indicative of a better-fitting model.

---

[7] PROC GLM performs complete case analysis omitting observations with missing variables. This can come back to haunt you, and you may want to recode missing variables to some number or character, so those observations contribute to your analysis.

[8] Just to keep things interesting, in SAS, the total sum of squares is referred to as sum of squares total (SST), between group or model sum of squares is referred to as sum of squares model (SSM), and within group or residual sum of square is referred to as sum of squares error (SSE).

[9] Sometimes referred to as the coefficient of determination, though not by me.

The next statistic you should look at is the coefficient of variation. In our presentation on MEANS and UNIVARIATE, we defined the coefficient of variation as σ/μ. In the setting of ANOVA it is the mean squared error (MSE) or error as a percentage of the overall mean.

$$\text{Coefficient of Variation} = \text{Root MSE / Weighted Mean}$$

While the calculation is a bit different, the interpretation is similar. It is a measure of how noisy the data is, with a larger value indicating more overall variation.

Now would be a good time to look at Levene's test for the homogeneity of group variances. As noted in the sample syntax above, Levene's test is requested in SAS as the "hovtest" option for the MEANS calculations in PROC GLM. The null hypothesis is that the variances are all equal[10] (which is one of the underlying assumptions for ANOVA), so you do not want to reject the null. A large p-value is what you are, in fact, looking for.[11]

If at this point, if we are satisfied that there is, indeed, some difference across group means, i.e., that group status is in fact an important consideration in our outcome variable, we will try to identify which group(s) is (or are) "more" different than the others.

We can get an initial sense of which group(s) differs from the others by a side-by-side comparison of boxplots. We can then get more precise information from the PROC GLM output titled Test Differences Between Means which lists the levels of the class variable and their μ and σ. The challenge is to pinpoint which of the differences are sufficiently meaningful from a clinical or public health perspective and also not likely to have been due to chance.

11.3.2 Multiple Comparisons

You might consider doing a test of statistical significance for the difference between each possible pair of groups. But you would be misguided. Recall that each such statistical comparison is subject to a 0.05 probability of type I error, i.e., rejecting your null hypothesis of no difference when in fact it is true. As the table below demonstrates, this type I error cost adds up very quickly.[12]

[10] The more groups in your class variable, the greater the likelihood that at least one of them will violate an assumption of normality or equal variance (homoscedasticity), although pooling the observations and increased numbers will buy you some flexibility.

[11] You should know that Levenes test is considered underpowered to detect differences in variances.

[12] The calculation for the true error rate is $1-(1-\alpha)^c$, where c is the number of comparisons.

We control the multiple comparison error rate through adjusting. In SAS, *LSMEANS* requests multiple comparison methods.

Number of comparisons	True error rate
1	0.05
3	0.14
6	0.26
10	0.40

ADJUST = TUKEY adjusts for all possible comparisons by dividing σ (traditionally 0.05) by the total number of all possible comparisons. *ADJUST = BONFERRONI* controls for preplanned comparisons by dividing σ by the number of comparisons you plan to conduct.

The syntax (included in the template above) is fairly simple:

```
lsmeans cat_var / pdiff=all adjust=tukey;
```

pdiff in the syntax above requests all pairwise p-values. You can similarly request a Bonferoni adjustment with *adjust=bon*.

TEA TIME

Sometimes in SAS, options can be indicated by shorthand versions of just their first letter. So, for example, rather than write out residual, you can just write r. This is very much not the case with ADJUST=TUKEY. If you were to write ADJUST=T, you would get all possible individual t-tests. You know, what you were trying to avoid in the first place.

11.3.2.1 LSMEANS

Interpreting the output of LSMEANS requires some thought, and we'll go through it in a bit more detail. You will see something like the following, which presents mock data for the example of the effect of clinic on diastolic blood pressure. We start by assuming that the results of the F-test for the effect of clinic on blood pressure were statistically significant.

This first result presents the means for the mean diastolic blood pressure reading for each clinic population. Just by looking at them, it appears that the clinics differed. Clinic 1 appears to have achieved a lower mean diastolic blood pressure reading. Our first step in interpreting the results of LSMEANS is to order the means. Here, they are 1<3<4<2 (Fig. 11.1).

We now look at the test for statistical significance for each pairwise comparison. Assume that we requested an appropriate adjustment, either Bonferroni or Tukey (Fig. 11.2).

Fig. 11.1 Group means

Least Squares Means

clinic	diastolic LSMEAN	LSMEAN Number
1	62.8571429	1
2	84.4285714	2
3	73.7142857	3
4	78.8571429	4

Least Squares Means for effect clinic
Pr > |t| for H0: LSMean(i)=LSMean(j)

Dependent Variable: pressure

i/j	1	2	3	4
1		<.0001	0.0062	0.0002
2	<.0001		0.0068	0.1365
3	0.0062	0.0068		0.1679
4	0.0002	0.1365	0.1679	

Fig. 11.2 Adjusted statistical significance of differences between means

Again, just by "eyeballing" the table, we can see that some comparisons were statistically significant. To more systematically interpret the table, we return to our ordered list from above. Now draw a line under each comparison that is *not* statistically significant. Here, the comparisons between clinic 3 and clinic 4, and between clinic 2, and clinic 4.

Proceeding in this way, we conclude that clinic 1 was statistically different from all the other clinics. Whether this difference is clinically meaningful is not a question LSMEANS or any other statistical test can answer for us.

11.4 Demonstration of One-Way ANOVA

Data set bp_drug contains systolic blood pressure measurements on 120 individuals each taking one of four drug doses.[13] We are interested in the effect of drug dose on lowering blood pressure. Our first step is to run some preliminary descriptive statistics on the data themselves.

```
proc univariate data=bp_drug;
var BP;
histogram / normal;
probplot;
run;
```

[13]This data set, as do so many in this text, comes from the SAS Institute.

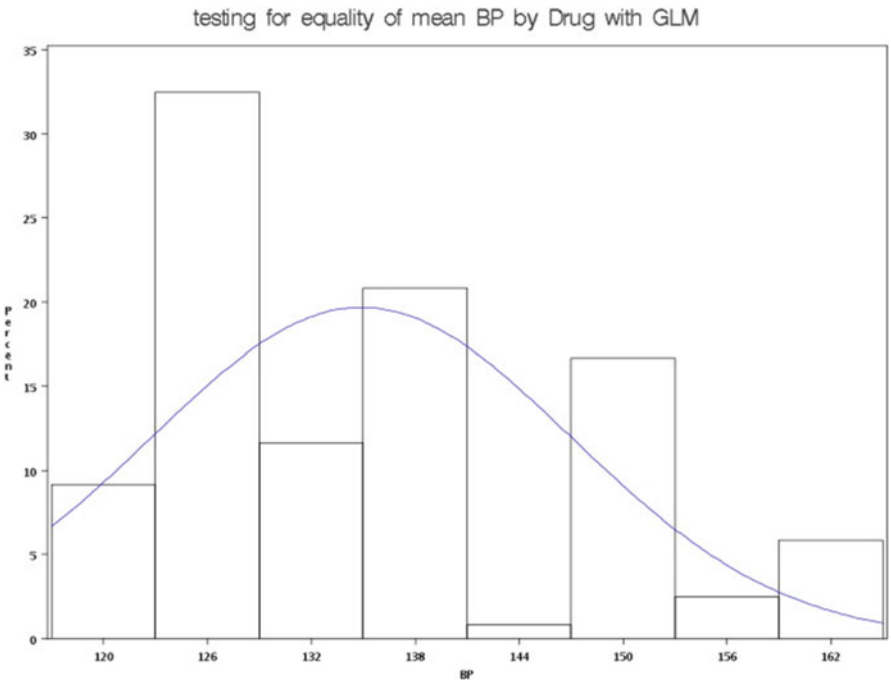


Fig. 11.3 Testing assumption of normality with a histogram

```
proc sort data=bp_drug;
by drug;
run;

symbol color = salmon;
   proc boxplot data=bp_drug;
      plot BP*Drug      / cframe  = vligb
                          cboxes   = dagr
                          cboxfill = ywh;
      title 'Boxplot Blood Pressure Drugs';
run;
```

The run returns the following results (Figs. 11.3 and 11.4).

We see from our univariate procedure that the mean systolic blood pressure for the entire sample was 134.9 which is higher than the upper limit of a desired systolic blood pressure of 120 mm Hg. You would have to rely on your own clinical knowledge or consultation with clinical folks to help evaluate this finding. The standard deviation for this mean is about 12 with a coefficient of variation of about 9% indicating some potentially important scatter of the data. The mean is not too very far off from the median of 132, and the skewness and kurtosis statistics do not appear problematic. Your histogram and probability plot may, though, cause you to question the underlying normality of these data.

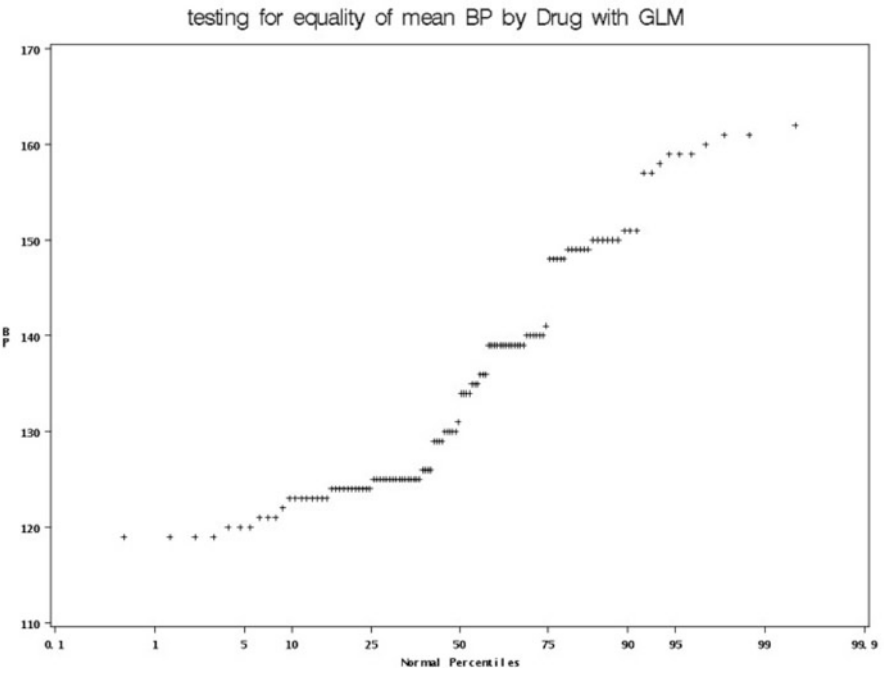


Fig. 11.4 Testing assumption of normality with a normal plot

The box plots indicate that there may be a meaningfully lower blood pressure associated with at least one of the drug doses (Fig. 11.5).

Our next step is to run PROC GLM to create a residual data set and further explore whether the assumptions for ANOVA are met.

```
options ls=75 ps=45;  /* page and line spacing options*/
proc glm data = bp_drug;
       class drug;
       model BP=drug;
       means drug / hovtest;
       output out=check r=resid   p=pred;  /* creating
                        an output data set called 'check'
                        'r ' is a keyword SAS recognizes as
                        residuals, 'p' recognized as
                        predictors*/
       title 'testing equality of mean BP by Drug Dose with
       GLM';
run;
quit;  /* have to quit out of GLM */

/* now run gplot on the 'check' dataset created above*/
Proc gplot data=check;
         Plot resid*pred / haxis=axis1 vaxis=axis2 vref=0;
              /* can leave out if ok with defaults*/
         axis1 w=2 major=(w=2) minor=none offset=(10pct);
         axis2 w=2 major=(w=2) minor=none;
         title 'plot residuals vs predictors for drugs';
```

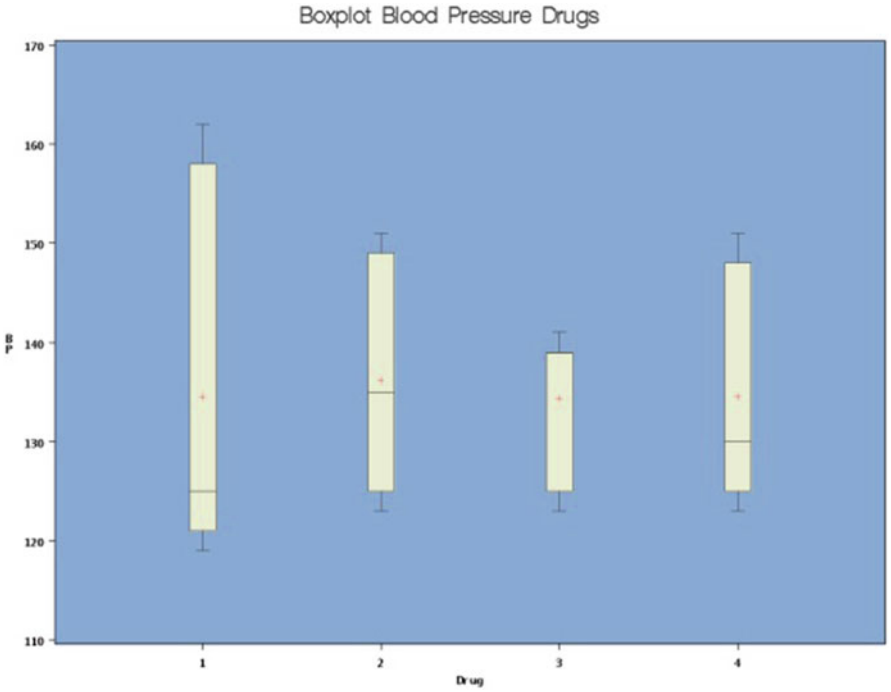


Fig. 11.5 Assessing group differences graphically with boxplots

```
run;
quit;

/* now run proc univariate to get histogram, normal plot,
           kurtosis and skewness on residuals*/
proc univariate data = check normal;
      var resid;
      histogram resid / normal;
      probplot;* resid / mu=est sigma=est color=blue w=1;
      title;
run;
```

Let's ignore the actual ANOVA for the time being, and first turn our attention to Leven's test for homogeneity of variances (Fig. 11.6).

The very small p-value indicates that we can reject our null hypothesis of homoscedasticity. It is likely then that variances varied across drug dose groups to an extent not supported by our underlying assumptions of ANOVA. Not, in fact, what we were looking for.[14] Let's see what our PROC GPLOT run of our residuals reveals (Fig. 11.7).

[14]I would not worry too much about this result. We buy ourselves some flexibility with larger numbers and more groups. The one assumption that requires strict adherence, though, is that of independent (uncorrelated or related) observations.

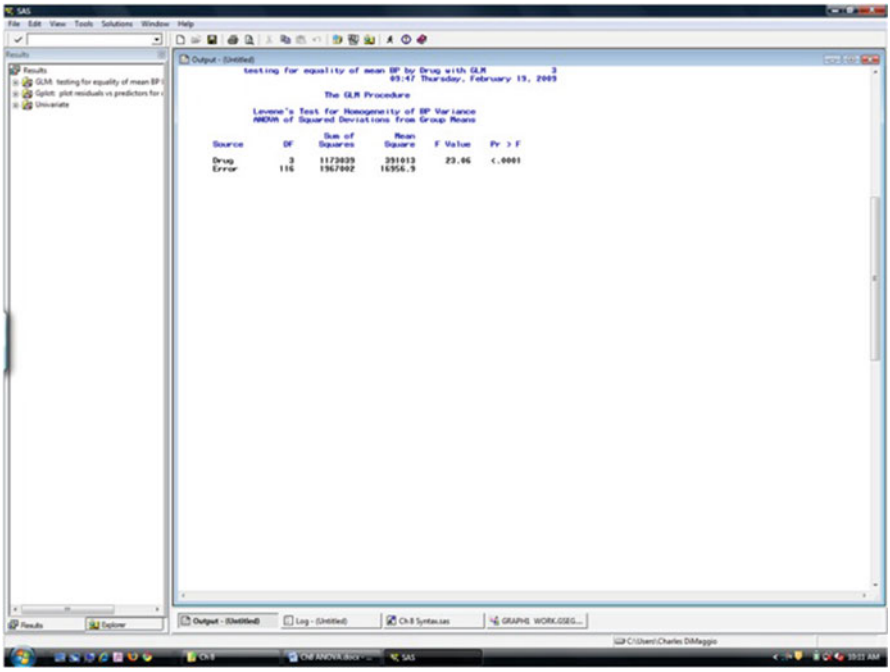


Fig. 11.6 Leven's test for homogeneity of variances

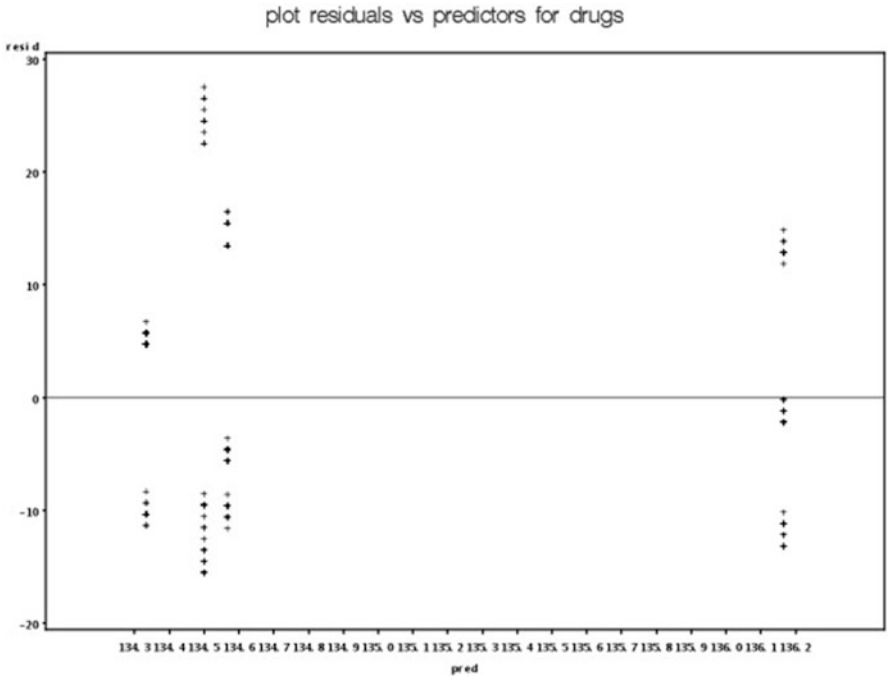


Fig. 11.7 Plotting the residuals

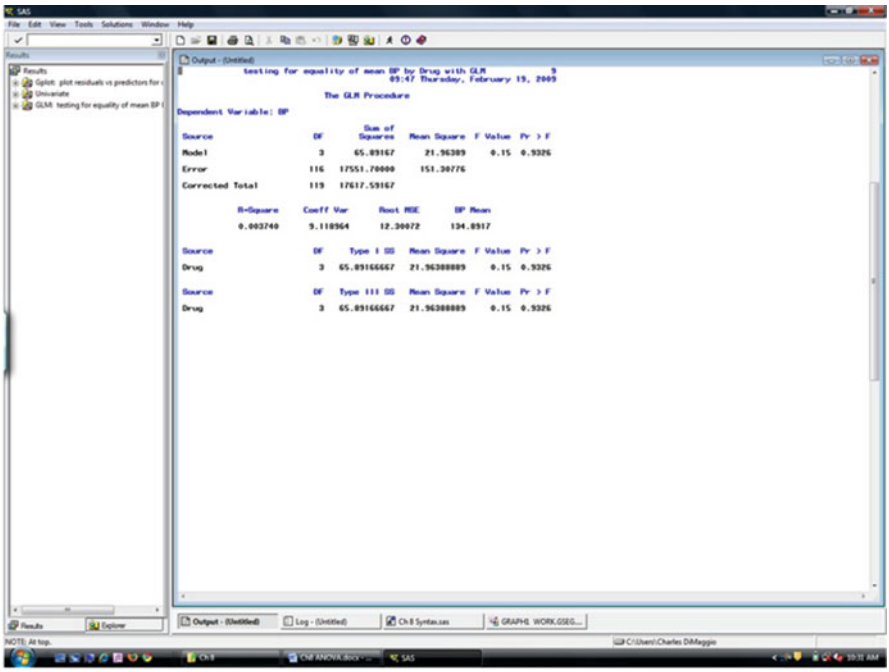


Fig. 11.8 PROC GLM ANOVA results

The data points for each group seem reasonably randomly dispersed around zero.

DON'T BLINK

In medicine, there is reference to so-called "eigenblink" diagnoses, those situations where one's initial almost instantaneous impression is most likely the correct one. This is a good approach to residual plots. Your first relatively quick impression is usually the correct one. Humans like to see patterns. If you spend too much time looking at residual plots, you may talk yourself into seeing patterns where none exist.

The PROC UNIVARIATE results generally reflect those of the data itself, which in the setting of a relatively straightforward one-way ANOVA is what we would expect. At this point, we would (if satisfied that our assumptions are reasonably well met) proceed with an evaluation of the actual ANOVA. This next screen presents the main results of our ANOVA (Fig. 11.8).

Based on the overall nonsignificant F statistic, it is unlikely that there are true underlying differences among the four drug dose groups. Our R^2 statistic indicates that group status explains very little of our total variance. At this point, you might

be rather disappointed in the effect of drug dose on blood pressure. But don't write this analysis off just yet. In our next section, we will see that perhaps something else is going on here.

11.5 Accounting for More than 1 Categorical Variable: n-Way ANOVA and Interaction Effects

Up until now, we have been considering the case of a one-way ANOVA, i.e., the effect of one categorical predictor variable on a continuous response variable. n-Way ANOVA refers to the case when we have more than 1, or n, categorical predictor variables.

As an example, say in our example of four antihypertensive drug doses, our patients could have had one of three different diseases. Our continuous outcome variable remains systolic blood pressure. Now, though, we have two categorical predictor variables: drug dose and disease. We are also likely interested in whether any particular combination of drug dose and disease interacts to affect blood pressure in perhaps an unexpected way.

Our model can be represented as

$$y_{ijk} = \mu + \alpha_i + \beta_j + (\alpha\beta)_{ij} + \varepsilon_{ijk}, \tag{11.5}$$

where

- y_{ijk} is the observed blood pressure for each subject
- μ is the overall population mean blood pressure
- α_i is the effect of disease i
- β_J is the effect of drug dose j
- $(\alpha\beta)_{ij}$ is the *interaction* between disease i and drug dose j
- ε_{ijk} is the residual or error term for each subject

We again assume that our outcome or response observations for each treatment are independently, identically, and normally distributed with approximately equal variances (homoscedasticity). Again, before conducting ANOVA, we would explore our data with descriptive statistics and examine our assumptions through residual analyses.

In contrast to our previous example, we now have two categorical or group variables (drug and disease) and we have included a statistical term for the combination of both drug and disease. This combination is an *interaction term*. We will discuss the epidemiological implications and approach to interaction shortly. For now, though, we will define interaction briefly as the situation where the relationship between a predictor variable and an outcome or response variable differs by different levels of a second predictor variable. So, here, we are interested in whether the effect of a drug on blood pressure varies in some important way depending on the presence or absence of one of three different diseases.

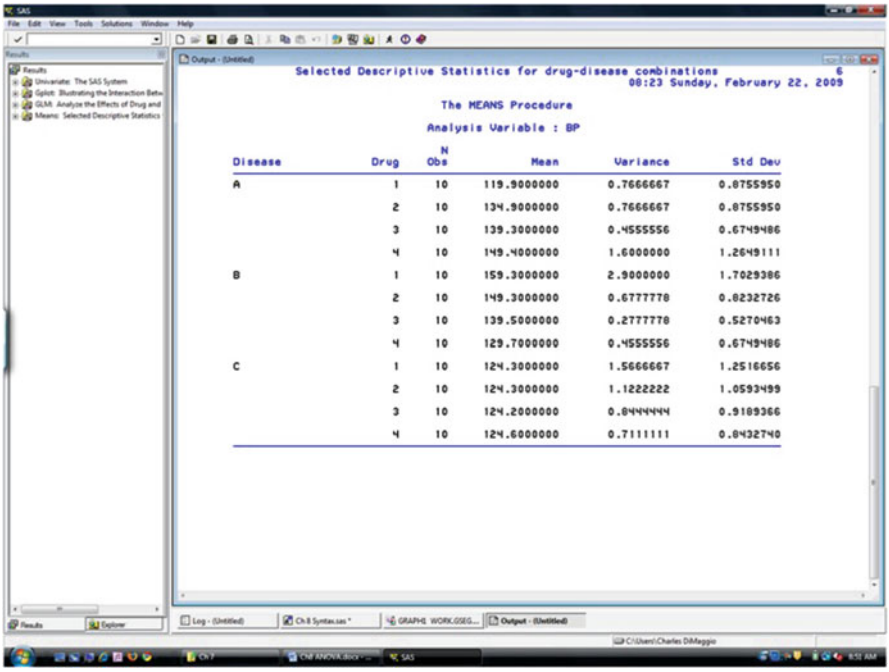


Fig. 11.9 The group means

We begin again with some descriptive statistics; here, a simple PROC MEANS of the 12 combinations of four drug doses and three diseases will suffice.

```
proc means data=bp_drug mean var std;
   class disease drug;
   var BP;
   title 'Selected Descriptive Statistics for drug-disease
   combinations';
run;
```

We see that in the setting of Disease A, drug level 1 produces the lowest BP. In the setting of Disease B, drug level 4 produces lowest BP. In the setting of disease C, the results are fairly homogenous (Fig. 11.9). We are, in fact, led to consider possible interactions between drug dose and disease.

An informative approach is to illustrate these potential interaction effects with a mean plot. This is a plot of our (continuous) blood pressure outcome variable on the *y*-axis vs. drug dose on the *x*-axis stratified by each of the three diseases (Fig. 11.10).

```
proc gplot data=bp_drug;
   symbol c=blue w=2 interpol=std1mtj line=1;
       /* interpolation method gives s.e. bars */
   symbol2 c=green w=2 interpol=std1mtj line=2;
   symbol3 c=red w=2 interpol=std1mtj line=3;
   plot BP*drug=disease; /* vertical by horizontal */
```

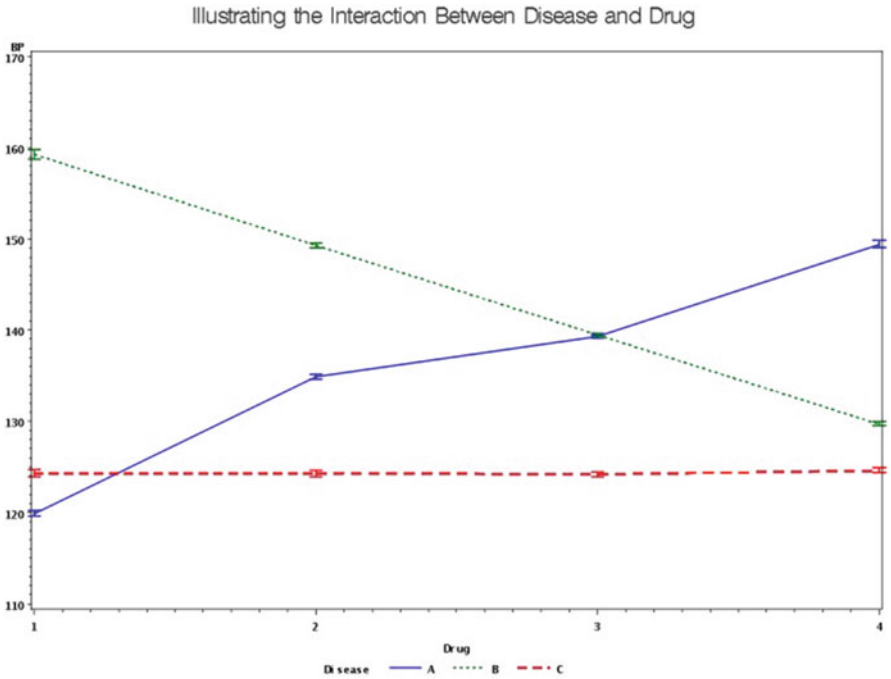


Fig. 11.10 Graphical assessment of interaction

```
   title 'Illustrating Interaction Between Disease and Drug';
run;
quit;
```

If the response is the same for each disease (no statistical interaction), we should see a set of more or less parallel lines illustrating the similar response of blood pressure (the *y*- or vertical axis) for the 4 different drug doses (*x*- or horizontal axis) for each of the three disease types.

Here, though, the responses are quite obviously not homogeneous (parallel). Blood pressure increases with increasing levels of drug in the setting of disease A, decreases with increasing level of drug for disease B, and remains at a fairly constant and low level for all dosing regiments for disease C.

Our next step, is to examine the statistical significance of the β coefficient for the interaction term (Fig. 11.11).

```
proc glm data=bp_drug;
   class disease drug;
   model BP=disease drug disease*drug;/*note interaction term*/
   title 'Analyze the Effects of Drug and Disease';
   title2 'Including Interaction';
run;
quit;
```

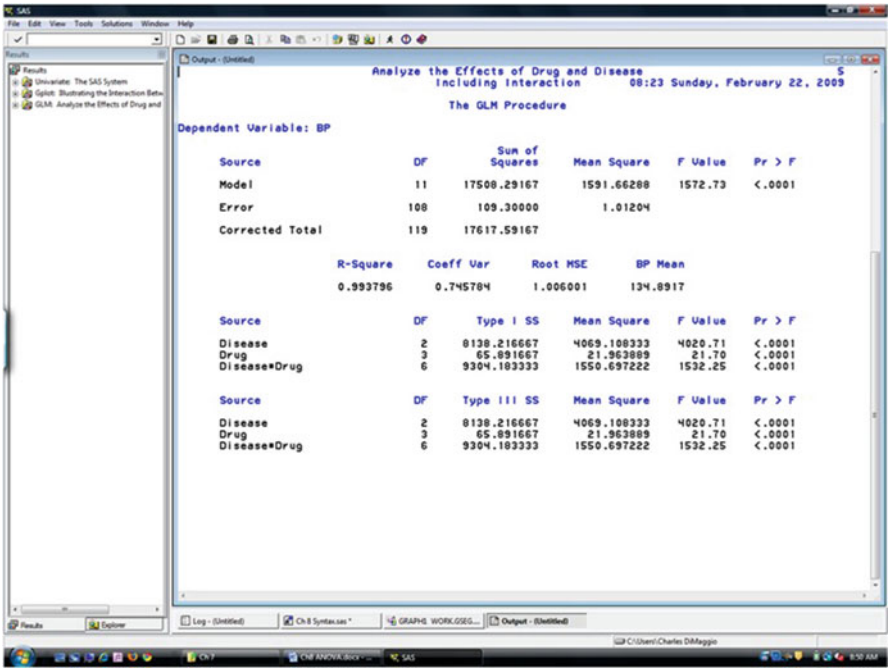


Fig. 11.11 Statistical assessment of interaction

First, look at the overall p-value for the model. If it is not significant, you may as well stop there. Here we have a model that is unlikely to be due to chance. Next, look at the R^2 statistic to get a sense of model fit or how much variance is explained by the model. Our model explains almost *all* the variation in the data.[15]

What, then, can we conclude from the results of this ANOVA? Only that the variables "matter" in that one treatment mean differs from at least one other treatment mean in a statistically significant fashion.

We next turn our attention to the interaction term. It is statistically significant, indicating statistical interaction between the two variables. What this means conceptually is that we cannot statistically consider the effect of the 4 drug levels without considering the disease. More practically, our model must retain both the variables that make up the interaction (whether they themselves are individually statistically significant or not). Our next step is to look at the results of our LSMEANS statement to assess the combinations of drug dose and disease for statistical significance (Fig. 11.12).

```
proc glm data=bp_drug;
   class disease drug;
```

[15]This is a remarkable change from our previous, one-way ANOVA. What a difference a variable makes.

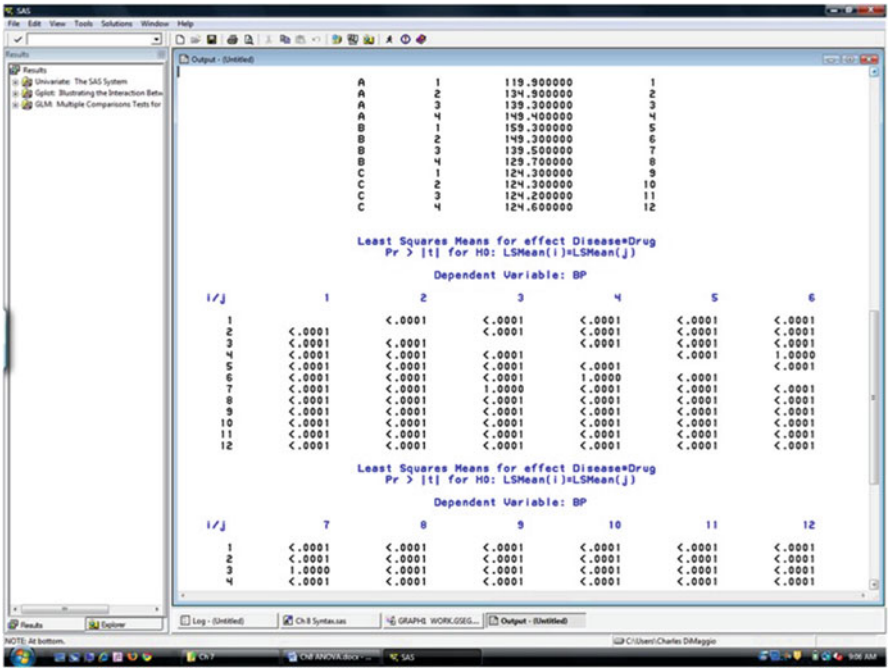


Fig. 11.12 Statistical significance of differences between means in the setting of interaction

```
    model BP=drug disease drug*disease;
    lsmeans disease*drug / adjust=tukey pdiff=all;
                /*looking at combinations*/
    title 'Multiple Comparisons Tests for Drug and Disease';
run;
quit;
```

Suffice it to say that the output for interaction terms tends to be voluminous and not entirely straightforward to interpret. Here, we see the 12 possible combinations of drug and disease.[16] From a purely statistical viewpoint, we would compare the systolic blood pressure means for all the different drug-disease combinations, identify those that were statistically significant, and draw conclusions on these (ostensibly) objective criteria.

But, from an epidemiological perspective, interaction is not quite so straightforward, which leads us to a (not so) short discussion of interaction and effect modification.

[16]There is a little trick to make sense of these kinds of results. First, order the means from lowest to highest. Then draw a line under the sets of means that are *not* statistically significantly different. The means without lines under them differ from each other.

11.6 Interaction and Effect Modification: An Epidemiological Perspective

The following discussion has little to do with SAS, but everything to do with an epidemiological appreciation of interaction and effect modification.

11.6.1 The Conundrum of Interaction

When an epidemiological measure of disease risk (either absolute or relative) behaves differently in the presence or absence of another variable, we typically say interaction is occurring. This is also sometimes referred to as *heterogeneity of effect.* That we can measure disease risk on either absolute scales, like risk differences, or on relative scales, like ratios, leads to a conundrum. Because there are two different measure of effect, there are two ways to describe interaction. And interaction may be present on one measures of effect, but absent on the other.

Additive interaction is based on absolute measures, like risk differences. We consider additive interaction to be present when the *absolute* measure of the risk of disease when two factors are present *differs from the sum* of the individual absolute risk measures.

$$\text{Additive interaction: } RD_{1,2} \neq RD_1 + RD_2$$

Multiplicative interaction is based on *relative* or *ratio* measures. We consider multiplicative interaction to be present when the relative measure of the joint risk or two risk factors being present differs from the *product* of the individual absolute measures:

$$\text{Multiplicative interaction: } RR_{1,2} \neq RR_1 \cdot RR_2.$$

We can illustrate these two types of interaction from the following example. The table below illustrates these two measures of interaction with an example of the association of stress and genetics with depression. Each cell contains the rates of depression in some population for people with or without a genetic marker and with or without a stressful life event.

	No stress	Stress
No genetics	10	17
Genetics	10	33

The risk difference associated with stress alone is $17-10=7$. The risk difference associated with genetics is $10-10=0$. To determine if additive interaction between stress and genetics is present, we first add up the individual absolute risk differences associated with each $(7+0=7)$ and then compare this expected

absolute risk estimate with the actual observed absolute risk estimate (23). Since the observed estimate far exceeds the expected, we can say there is additive interaction (or interaction on an additive scale).

On the multiplicative scale, the relative risk for stress alone is $17/10 = 1.7$, and that for genetics alone is $10/10 = 1$. To test if multiplicative interaction between stress and genetics is present, we multiply the individual relative risk differences associated with each ($1 \times 1.7 = 1.7$) and then compare this expected relative risk estimate with the actual observed relative risk estimate (3.3). Since the observed estimate exceeds the expected, we say there is multiplicative interaction (or interaction on a multiplicative scale).

So far, so good. Interaction on one measurement scale implies interaction on the other scale. But the plot thickens It turns out, quite to the concern of epidemiologists, that the *absence* of interaction on one scale, rather than implying the absence of interaction on the other scale, is almost invariably accompanied by the *presence* of interaction on the other scale. So, if there is no *additive* interaction, there is very likely *multiplicative* interaction, and vice versa. The following rate data on the potential interaction between life events and intimacy problems on the risk of depression illustrates this concept.

	No life event	Life event
No intimacy problems	1	3
Intimacy problems	10	32

The referent value for the absolute risk associated with intimacy problems is $3 - 1 = 2$, and the absolute risk associated with life events is $10 - 1 = 9$. The expected *joint* absolute risk is then $2 + 9 = 11$. We see, though, that the observed joint risk is 31, indicating interaction. If, though, we look at the multiplicative scale, things appear quite different. The referent value for the *relative* risk associated with intimacy problems is $3/1 = 2$, and the relative risk associated with life events is $10/1 = 9$. The expected *joint* relative risk is then $2x10 = 30$. When we compare this to the observed joint relative risk of 32, we find little or no interaction.

Here then, is the conundrum: how can the two risk factors interact to cause more disease than expected while at the same time *not* interact and cause only the level of disease expected from the individual action of each alone? The answer is, of course, that they can't. But which scale is "right"? The answer to that question requires, as if often the case, that we begin at some basic concepts and work forward.[17]

[17]Regression models themselves, which we will introduce in coming chapters, are also either additive or multiplicative and therefore subject to this ambiguity. Linear models are additive, and logistic models are multiplicative (don't let those plus signs in a logistic model fool you; on the log scale what looks like addition is actually multiplication).

11.6.2 Components and Causes

We need to start with that most fundamental issue of science: separating coincidence from causality. Say your phone rings, then your tea kettle whistles. Most reasonable folks would see no connection other than coincidence. On the other hand, say a stranger approaches your home and your dog barks. You will likely have an intuitive sense that these two events are causally related. We might reasonably label the sequence causal *if the subsequent event would not have occurred without the prior event having occurred*. The epidemiological concept of the risk factor rises naturally from this concept.

A risk factor is the antecedent event without which some outcome would have occurred. To determine if a risk "caused" a disease, we compare what occurs when the risk factor is present, to what occurs when the only thing different is that the risk factor is not present.[18] Importantly, we recognize that life (and disease) is complicated and that there is likely more than one antecedent necessary for some outcome to occur. In the dog barking example, we assume that you have a home, that there is a window through which the dog might see the stranger approaching, etc.

A cause that consists of multiple components leads to the idea of "INUS" causes of disease: *I*nsufficient but *N*ecessary components of *U*nnecessary but *S*ufficient causes. The components of a cause are Insufficient in that they need other components and Necessary in that the disease will not occur without all the components being necessary. The causes themselves are Unnecessary in that they are not the only possible constellation of components that will result in disease, but Sufficient because the particular constellation of components will result in disease.

Because a cause consists of component risks, causal relationships are context dependent. The strength of any risk factor is relative to and dependent on the presence or absence of its causal partners. This has an interesting interpretation from a population perspective. As put by Susser and Schwartz, if the causal partners of the risk factor are rare, the people exposed to the risk factor will rarely develop the disease. Basically the strength of the association of a risk factor of a disease depends on the prevalence of it's component risk factors in the population under study.

Take, for example, the situation of neural tube defects in children. They are associated with both a genetic defect and low levels of maternal folate. In a population where all women have the genetic defect, but few have a low folate, diet will be the most important risk factor for disease since every woman with low folate who gives birth will have a child with a neural tube defect. By contrast, in a population where folate is ubiquitous, but the genetic defect is rare, the gene will appear to be the most important risk factor. We tease out these relationships by looking at different populations.

[18] Of course, this so-called "counterfactual" set of events cannot actually occur in nature. Epidemiology consists in large measure of study designs that try to mimic it.

With that healthy pre-amble, we can now consider how variables in individuals and populations produce, mediate or modify the exposure-disease relationship. There are five basic potential relationships between an exposure of interest and another variable or risk factor:

1. Independent risk factor–Causes disease through a causal pathway different than that of the exposure of interest (a different causal mechanism).
2. Antecedent–Precedes the exposure.
3. Confounder–An alternate risk factor for the disease, but associated with the exposure of interest.
4. Mediator–(Also) A risk factor for the disease but (unlike a confounder) does not provide an alternate explanation for disease
5. Causal partners—Other component members of a causal mechanism that combine with exposure and can result in synergy or interaction.

11.6.3 Usefulness of the Additive Model

Viewed from the framework of component causes, synergy, or interaction is the underlying process for how *any* cause results in disease at the individual level.[19] We may reserve the terms "statistical interaction" or "effect modification" for how we try to capture this idea of synergy, which we often do through statistical measures of interaction, as in our blood pressure example above.

Moving forward from the idea of component causes working together, Darroch [3] and Rothman and Greenland (1998) proposed a solution to the conundrum of interaction. The answer lies in the phenomenon of *parallelism*, which we define as the presence of individuals in a study population who can only develop disease from one or the other of the two causes. We can briefly walk along this line of reasoning.

We begin (again) by considering two risk factors for a disease, A and B. We now include a set of factors, U, to stand in for all the unknown factors that go into disease occurrence. These three factors may result in disease in 4 possible ways:

- R_{ABU}—the risk of disease from the interaction of A and B
- R_{AU}—the risk of disease from A
- R_{BU}—the risk of disease from B
- R_U—the "background" experience where disease occurs in the absence of either A or B

To determine if the observed R_{ABU} exceeds what we might expect if the two risks did not interact, we subtract out R_{AU} and R_{BU} and then add back R_U which we subtracted twice. Via a little algebra, if the two risk factors are biologically

[19] Although interaction and effect modification are inherently ambiguous, they are population-level phenomena, as opposed to biological interaction, which occurs at the individual level when the effect of one variable depends on the presence of another.

independent, then $R_{ABU} = R_{AU} + R_{BU} - R_U$. In terms of risk differences: $(RD_{AB} - RD_U) = (RD_A - RD_U) + (RD_B - RD_U)$ In terms of relative risks, $(RR_{AB} - 1) = (RR_A - 1) + (RR_B - 1)$. Any excess risk beyond these inequalities is due to interaction.

Let's work through a brief example involving cancer risks due to smoking and asbestos exposure.

	No asbestos	Asbestos
No smoking	1	5
Smoking	10	50

We set up our biological independence equality as $50 - 1? = (10 - 1) + (5 - 1)$, and since $49 \neq 13$ we conclude that the smoking and asbestos interact to cause more cancer than would be expected if either were present alone. We can say further that $49 - 13$ or 36 of every 50 cases (72%) of cancer when both smoking and asbestos are present are due to the interaction between them.

The procedure is the same if the results are on a relative scale. Consider the following table or relative risks for cancer due to smoking and asbestos.

	No asbestos	Asbestos
No smoking	1	3.1
Smoking	6.9	13.6

We test the equality $13.6 - 1? = (6.9 - 1) + (3.1 - 1)$, and since $12.6 \neq 8$, we conclude (again) that there is interaction and that $(12.6 - 8)/13.6 = 4.6/8 = 34\%$ of the cases when both risk factors are present is due to interaction.[20]

The bottom line is that biologic interactions must be measured on an additive scale, because it is based on partitioning the counts of cases into 4 categories (R_{ABU}, t R_{AU}, R_{BU} and R_U) that make causal sense. Multiplicative models (such as logistic regression) involve transformations that make these counts and categories unavailable.

11.6.4 A Final Thought on Interaction in Epidemiological Studies

So how, practically, should we approach interaction in epidemiological studies? For me, a graphical assessment remains a useful informative approach. As we did with

[20]Note that if we looked at these data on a multiplicative scale, then $3.1 \times 6.9 = 21.4$, and since $13.6 < 21.4$ the presence of both risk factors actually results in less disease risk than we expect.

the mean plot in our blood pressure example above, plot your response variable on the Y axis vs. presence or absence of your second factor (or varying levels of the second factor) on the X axis.

As for modeling interaction terms, the key is to address biological interaction as an additive phenomenon. Categorize the two potential interaction variables into a factorial design, where 11 represents the presence of both, 10 and 01 the presence of one or the other, and 00 the absence of both. Then test those risk estimates for a departure from additivity as we did above.

And what about tests for the statistical significance of interaction terms? While ambiguous, they remain an accepted approach. So when conducting an n-way ANOVA in which you believe interaction may be present, first look at the test for interaction in the ANOVA output to decide whether there is interaction between the factors In general, if there is no interaction between the factors, the tests for the individual factor effects can be considered in the output to determine the statistical significance of these factors. As we have seen in our example, though, if there is interaction between the factors, the tests for the individual factor effects might be misleading due to masking of these effects by the interaction.

Just know that you are addressing statistical, not necessarily biological concerns. While the above guidelines for dealing with *statistical* interaction are well accepted, the most appropriate *epidemiological* approach to interaction is a priori , conceptual, and informed by subject matter expertise. Think about it during data collection and consider scientifically plausible interactions. By contrast, if you just go looking for interactions, you may well find them, but they may be artifactual and perhaps even misleading. Use knowledge and evidence to guide your statistical approach.

Problems

11.1. PROC GLM

In a previous chapter, you examined the mean birth weight for each borough of New York City. You will now examine this question more formally with the tools of ANOVA. Using the infants data set you created previously, write the syntax to conduct an ANOVA testing the effect of borough on birth weight. Request Levene's test for the homogeneity of variances. Create an output data set of residuals.

What is the p- value associated with Levenes test for homogeneity of variances? How do you interpret this?

11.2. PROC GPLOT

Now run PROC GPLOT on the residual data set you created. Plot the residuals against your predictor or explanatory group variable.

What is your interpretation of the resulting graph of residuals?

11.3. PROC UNIVARIATE

Run a PROC UNIVARIATE on your residual data set. Request a histogram and a normal probability plot. Examine the skewness and kurtosis statistics.

What is the mean? Is this consistent with the underlying assumptions for ANOVA? What is the kurtosis statistic? How do you interpret this result?

11.4. LSMEANS

Re-run your PROC GLM and compare the mean birth weight for each borough. Request and adjust for all possible pair-wise comparisons.

What is the lowest mean birth weight? Is this statistically significantly different from the other boroughs? What graphical procedure would be helpful to compare the mean birth weights across boroughs?

Chapter 12
Correlation

Abstract In this chapter, we consider the (possible) relationship between two continuous variables. We might, for example be interested in how blood pressure varies with weight, or if the number of years of schooling is associated with cholesterol levels. The SAS syntax for examining these kinds of relationships is fairly simple and straightforward, but the interpretation demands some caution.

12.1 Assessing Correlation

Correlation is a linear relationship between two continuous variables. The relationship could be direct or positive, i.e., when one variable increases (or decreases) the other variable changes in the same direction as well, or it could be indirect or negative, i.e., when one variable increase (or decreases), the other variable changes in the opposite direction.

The first step in assessing a potential correlation between two variables is graphical, and PROC GPLOT is the most useful tool in this regard. In assessing such scatter plots, you generally don't want to spend too much time looking at them. Your first, or *Eigenblink*, assessment is usually correct. You will examine the plots for the presence of linear (positive or negative), curvilinear, cyclical, or no relationship.

So, for example, in the following graphic, figure 1 represents a positive or direct linear relationship between two variables, figure 2 a curvilinear relationship, figure 3 a cyclical relationship, and figure 4 no relationship at all (Fig. 12.1).

Of course, the actual plots you will see in practice will not be so clear. Lets take a look at an example to get a sense of how to interpret scatter plots to help you determine relationships between continuous variables. A data set, which I have titled "work.corr" contains 33 observations that contain a persons age, their height,

C. DiMaggio, *SAS for Epidemiologists: Applications and Methods*,
DOI 10.1007/978-1-4614-4854-9_12,

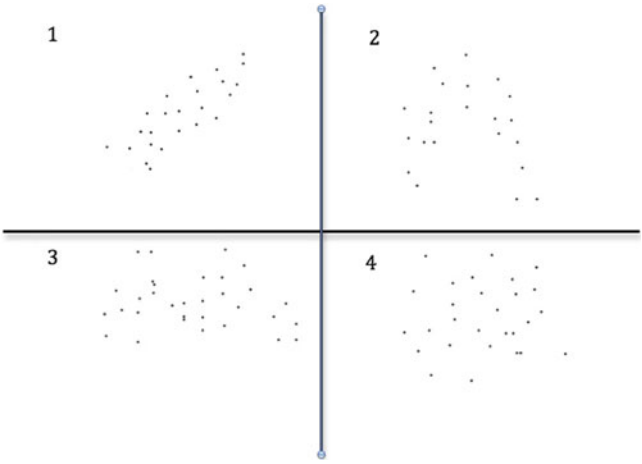


Fig. 12.1 Some scatterplot patterns

and two continuous outcome variables related to lung capacity.[1] Let's first look at the correlation between age and height (Fig. 12.2). The GPLOT syntax is:

```
proc gplot data=corr;
title1 'direct correlation ?';
plot age*height;
run;
quit;
```

What are your thoughts on this scatter plot? My initial impression is that there is a weak, but possibly, positive linear relationship between age and height.

Using similar syntax, we produce the following scatter plot of age with the outcome2 variable. Again, while not clear cut, there appears to be a negative linear relationship (Fig. 12.3).

Just looking at scatterplots is an informative way of assessing correlation. We will, though, want to quantify any apparent relationship. In the next section, we will consider a statistic that can more formally assess these potential relationships.

12.2 Assessing Correlation Using PROC CORR

One of the most common statistics used to evaluate the linear relationship between normally distributed continuous variables is the Pearson correlation coefficient, represented by a lower case *r*.

[1]Though I have substantively manipulated these data, I originally found these observations online and have since been unable to re-locate the site to properly attribute. My apologies to the author. If they are your data, do let me know.

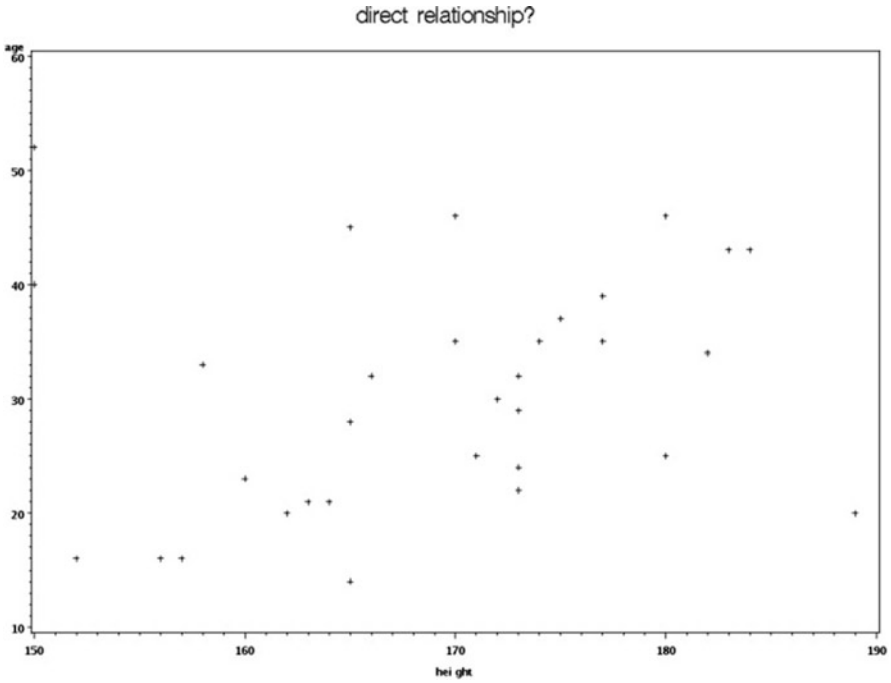


Fig. 12.2 Scatterplot age and height

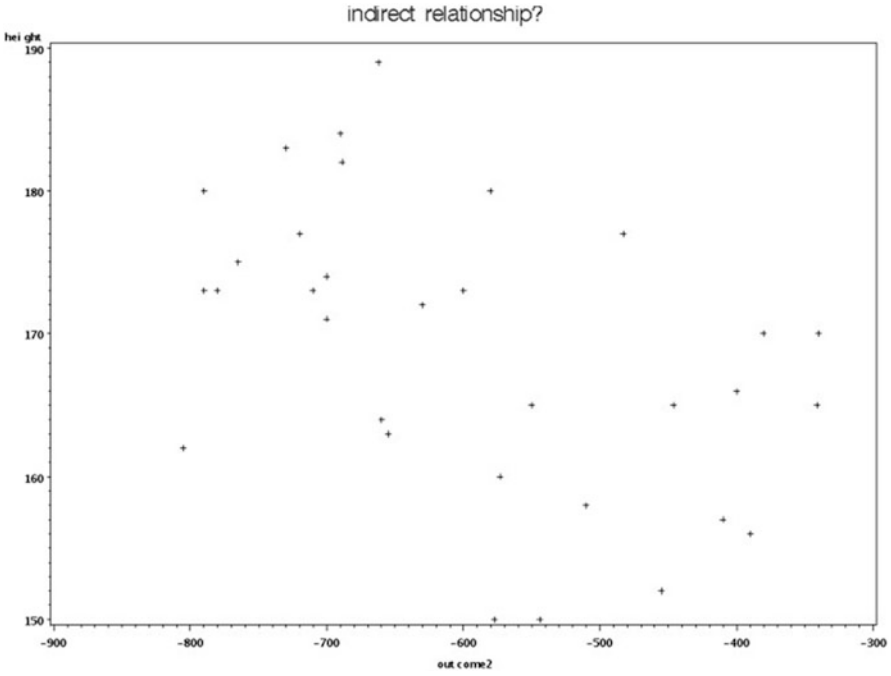


Fig. 12.3 Scatterplot age and lung capacity variable

The Pearson correlation coefficient (r) is defined as the sum of the products about the mean of two variables divided by the square root of their sum of squares:

$$r = \sum(x_i - \bar{x})(y_i - \bar{y}) / \sqrt{\sum(x_i - \bar{x})^2 (y_i - \bar{y})^2}. \quad (12.1)$$

The formula returns a value between −1 and 1 to reflect the strength of negative or positive relationships. The closer the coefficient is to zero, the weaker the relationship.[2] How should you interpret the strength of a correlation coefficient? There are any number of rules of thumb floating around of which I've found the most useful to be that a correlation of less than 0.30 reflects little or no relationship between the variables. Correlations of 0.5 or higher might indeed be important. Between 0.3 and 0.5 fall in a grey zone: not very strong, but interesting.[3]

Another useful approach is to square the r value to get r^2 which is our old friend from the previous chapter, the coefficient of determination, and tells us how much of the variation in one variable is explained by knowing the other variable. So a correlation coefficient of 0.5, would mean knowing one variable would allow us to account for or explain 25 % of the variability in the other variable.

HOLD THE LINE

As you will see below, correlation coefficients must be evaluated cautiously. They may be spurious, they may be masking a non-linear relationship, and they may be unduly influenced by an outlier observation. Graphing and considering descriptive statistics will be most helpful.

A correlation coefficient is a convenient summary of the relationship between two variables, but a few caveats are in order. First and perhaps most important: correlation does not mean causation. So for example, weight and height are strongly correlated, but weight does not cause height or vice versa. The following scatter plot represents the relationship between the yearly average Dow Jones Industrial Average and the yearly asthma hospitalization rate per 10,000 for children aged between 1 and 17 for the years 1980–2000. There certainly appears to be some kind of potential linear relationship (Fig. 12.4).

I calculated the Pearson correlation coefficient for these two variables and got an $r = 0.45$ indicating a potential correlation. So, does wealth cause asthma? It is hard to imagine a biologically plausible explanation. Notice that this is not a statistical issue; it is an epidemiological and substance matter issue. Those lessons

[2]Generally, if both variables consist of positive values, or both consist of negative values, you will get a positive correlation coefficient. If one consists of positive values and the other of negative values, you will get a negative correlation coefficient.

[3]Hinkle, Wiersma, and Jurs (1988) Applied Statistics for the Behavioral Sciences, 2nd edn., Houghton Mifflin Co.

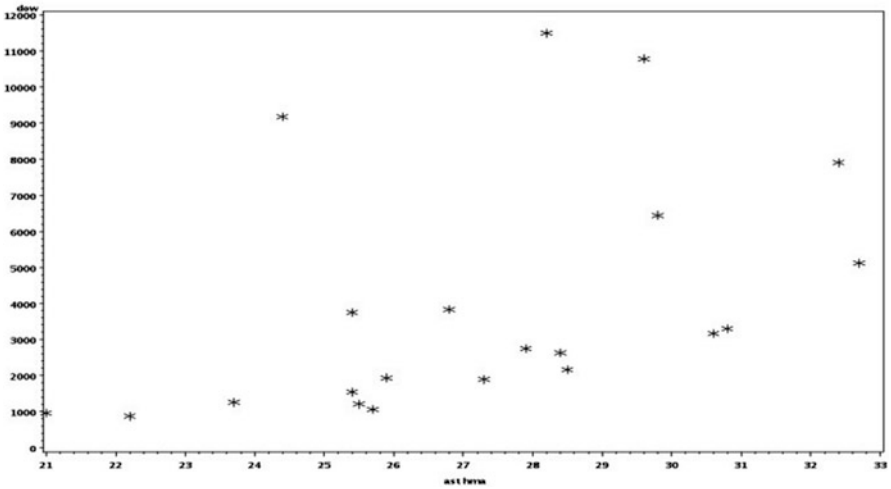
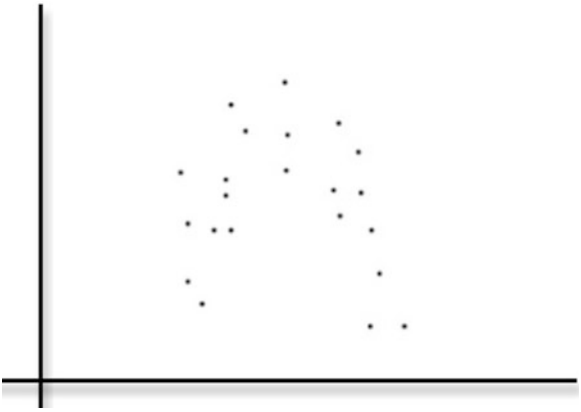


Fig. 12.4 Scatterplot Dow Jones industrial average and asthma

Fig. 12.5 A non-linear relationship



you learned about ecologic fallacy, bias, and confounding will help you explain this likely spurious correlation.[4]

A second caution: a correlation coefficient only measures *linear* relationships. If the relationship if curvilinear or cyclical, the result is suspect. The following set of data may very well return a correlation of close to zero, but it does not mean there is no relationship between the variables (Fig. 12.5).

Lastly, be on the lookout for outliers or extreme values that can push a summary measure like the correlation coefficient in one direction or another. The one lone

[4]This issue is so common it even has its own journal: The Journal of Spurious Correlations. (http://www.jspurc.org/).

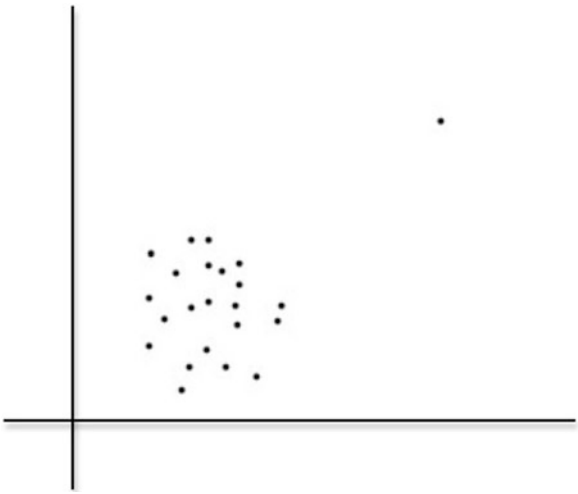


Fig. 12.6 An influential observation or outlier

value in the scatterplot below will influence the correlation coefficient to return a value close to one, when there is in fact no relationship between the variables (Fig. 12.6).

From this discussion, you may come to the conclusion that it is usually a good idea to examine your variables graphically and with descriptive statistics before calculating a summary measure like a correlation coefficient. You would be correct in this assumption.

The SAS code for correlation is the fairly simple and appropriately named PROC CORR. After invoking the procedure and specifying the data set, the rank option will order your correlations from high to low. You then specify a numeric variable with a VAR statement and the variable with which you want to pair it with a WITH statement[5]:

```
PROC CORR data=work RANK;  /* rank orders correlations from
  high to low */
        VAR predictor1;
        WITH outcome_var;
         Title correlation of outcome with predictor;
Run;
```

Lets run this procedure on the age, height and oxygen consumption data from the previous section.

```
proc corr data=corr rank;
var age;
with height;
run;
```

[5]If you want a correlation matrix for a number of variables, just omit the WITH statement.

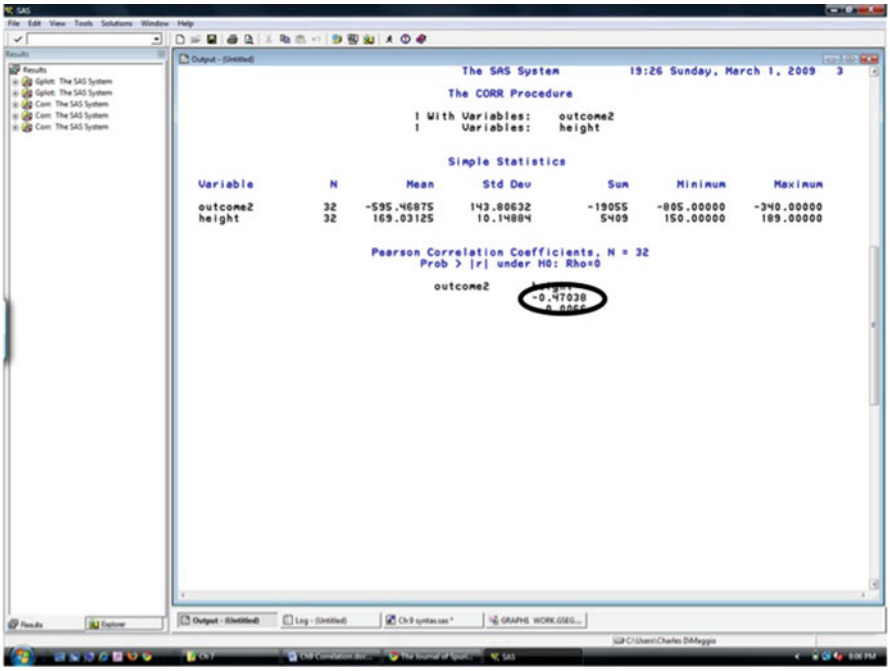


Fig. 12.7 PROC CORR results

```
proc corr data=corr rank;
var age;
with outcome2;
run;
```

We find that the only correlation that rises to potential importance is the inverse relationship between height and outcome2, which returns an $r = -0.47$.

I've circled the relevant result in the screen capture above (Fig. 12.7).

Correlation coefficients are useful, and we see them a lot in public health research. There is no denying, though, that regression is the most popular approach to modeling continuous data. We turn our attention to linear regression in the upcoming chapters.

Problems

Ecological studies are the most common epidemiological setting for the use of a correlation coefficient. The data file **ch12exercise.xls** consists of the number of pedestrian injuries and census characteristics of ZIP code tabulation areas (ZCTA) in Nassau County, New York. The variable NUMINJ is the number of injuries that

occurred in the ZCTA. TOTPOP is the total population for the area. PERBLACK and PERHISP refer to the proportion of the population that is African American and non-White Hispanic, respectively. MEDHSINC is median household income and PCI is per capita income. The data set is in Microsoft Excel format. The following exercises walk you through a typical analysis using correlation coefficients. You will see that the calculation of the pearson correlation coefficient is the last, and by no means the most informative, step in a series of analyses that call on procedures we have already covered.

12.1. Read in the Pedestrian Injury Data Set
Read the data into SAS. Explore it by running PROC CONTENTS.

12.2. Print Out the Data
Print out the data set. Title it Printout of Pedestrian Injury Data. Limit the print out to the variables for total population, number injured, percent African American population, percent Hispanic population, median household income, and per capita income. Use the name of the ZCTA (variable NAME) to identify the observations.

12.3. Create an Injury Rate Variable
Create a rate variable for pedestrian injures based per 1,000 total population.

12.4. PROC UNIVARIATE
Run a PROC UNIVARIATE analysis of the injury rate, percent African-American, percent Hispanic, median household income, and per capita income. Identify the observations by the name of the community (ZCTA). Create histograms and probability plots for the variables.

What was the average number of injuries per 1,000 population for a Nassau County ZCTA community during the observation period? Which community had the highest injury rate? What was it? Which community had the highest proportion of Hispanic residents and which had the second highest? Which community had the lowest median household income and which had the second lowest? Which two variables appear to be least likely to be normally distributed?

12.5. Log Transformation
Log transformations are helpful in addressing issues of normality. Use the log( ) function[6] to create two new variables based on the two non-normal variables you identified in exercise 12.4 Run PROC UNIVARIATE on these two new variables. Do they now appear more normally distributed?

12.6. PROC GPLOT
Use PROC GPLOT to create scatter plots examining the graphical association between injury rate and the appropriate variables for African-American and Hispanic populations in a community, median household income and per capita income:

```
/*set up the options for your plots and define axes*/
Options ps=50 ls=64;
```

[6]Use the SAS Help feature for log() to see how functions are used in SAS.

```
Goptions reset=all gunit=pct border
Fontres=presentation ftext=swissb;
Axis1 length=70 w=3 color=blue label=(h=3)
value=(h=3); Axis 2 length=70 w=3 color=blue
label=(h=3) value=(h=3);
```

Invoke the above options and axes by including the following line after your plot statement:

```
/vaxis=axis1 haxis=axis2
```

Which of the 4 plots seems to most demonstrate a relationship with injury rate? How would you describe that plot?

12.7. PROC CORR

Calculate the Pearson correlation coefficient for the relationship between injury rate and the appropriate variables for African-American and Hispanic populations in a community, median household income, and per capita income.

What variable was most strongly correlated with pedestrian injury rate in a Nassau County community? What was the relationship between income in a community and pedestrian injury rate? Sum up, in one or two sentences, your conclusions based on these results.

Chapter 13
Linear Regression

Abstract With correlation, we explored the linear relationship between 2 continuous variables. With simple linear regression, we seek to further define and describe the relationship between a single continuous predictor variable and a continuous outcome variable. Later, we will consider the more common situation of multiple linear regression where there are more than one predictor variables, some continuous some categorical.

13.1 Introduction to Regression

The term regression comes from Galton, who looked at the relationship between the heights of parents and their children. He noticed that the children of tall parents were closer in height to the population average than to the height of their own parents. He called this observation regression toward mediocrity. We usually refer to regression to the mean.

In regression analysis, we are interested in assessing the role of a predictor variable (x) that is assumed to be fixed, in explaining the variability of an outcome variable (y). We assess this relationship by sampling from some population to estimate the model:

$$y = \beta_0 + \beta_1 x + \varepsilon, \tag{13.1}$$

where

- β_0 is the outcome when the predictor y is 0.
- β_1 is the slope (the "rise over the run"), or amount of change in the outcome y per unit change in the predictor x.
- ε is the error term (defined as it was with ANOVA).

We estimate the relationship by using the method of least squares which sets the best fitting line through our data points, i.e., a line that comes as close as possible,

C. DiMaggio, *SAS for Epidemiologists: Applications and Methods*,
DOI 10.1007/978-1-4614-4854-9_13,

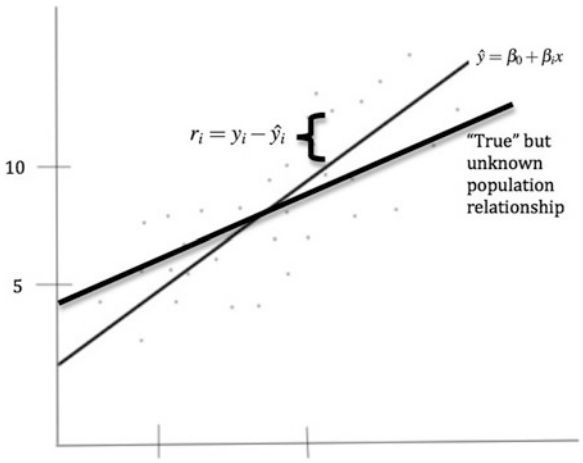


Fig. 13.1 Population relationship between two variables vs. the regression line

on average, to all the data points by minimizing the deviation of the distance from the line to the data points in the y direction (Fig. 13.1).

The distance from the estimated line to the data point is $y - \hat{y}$. This is the distance we want to minimize. Since positive and negative values will cancel each other out, we square the terms.[1] A little calculus takes care of minimizing the sum of the squared terms, $\Sigma(y - \hat{y})^2$.

There are then some fairly simple calculating formulas for the terms in a simple linear regression model:

$$\hat{\beta}_0 = \bar{y} - \hat{\beta}_1 \bar{x} \tag{13.2}$$

$$\hat{\beta}_1 = \frac{\Sigma(x_i - \bar{x})(y_i - \bar{y})}{\Sigma(x_i - \bar{x})^2}. \tag{13.3}$$

The confidence interval around $\hat{\beta}_1$ can be constructed as $t_{\alpha/2,n-2} * se$. And a hypothesis test for $\hat{\beta}_1$[2] is $\hat{\beta}_{1/se}$.

Our baseline, or null, model is that there is no relationship between x and y and that in fact just knowing the mean or average value ($\hat{y}$) will give us more or better information about y than x does. This is analogous to the model $H_0 : \beta_1 = 0$, and in simple linear regression, we test this null hypothesis that the slope of the regression line is not statistically significant from zero.

Even at this basic level, linear regression allows us to do a number of interesting things. We could compare 2 regression lines and test the hypothesis that the slopes are different from each other, or construct a confidence interval around the

[1] If you've been reading the chapters in sequence, you should by now be starting to appreciate how squared terms are one of those basic statistical concepts we see again and again.

[2] If you insist on doing hypothesis tests...

difference between the two slopes. We could use standard errors to estimate the precision of our estimated y values based on the value of $\bar{y}$ given some value of x, or (with more error) the precision of an individual estimate of y.

13.1.1 Variance Perspective of Regression

As we discussed in the chapter on ANOVA, the general linear model tells us that linear regression is part of the same family of models from which ANOVA arises.[3] This set the foundation for us to look at regression in terms of partitioning the total variance in the data (the total sum of squares or SST) into that explained by our model (model sum of squares or SSM) and unexplained variability (error sum of squares or SSE):

- Total variability (TSS) distance from data points to mean line:

$$\Sigma(y_i - \bar{y})^2.$$

- Explained variability (SSM) distance from regression line to mean line:

$$\Sigma(\hat{y} - \bar{y})^2.$$

- Unexplained variability (SSE) distance from data points to regression line:

$$\Sigma(y_i - \hat{y})^2.$$

As in ANOVA, if it's a good model, we expect more explained than unexplained variability and the proportion of the total variability explained by our regression model to be closer to 1 (Fig. 13.2).

The assumptions underlying linear regression are also the same as those we discussed with ANOVA:

1. Independence—first and foremost. Observations are not related and do not influence each other.
2. Linearity. There is a true, underlying linear relationship between the variables.
3. Normality. At any given value of predictor (x) the response variable (y) is normally distributed.
4. Homoscedasticity or homogeneity of variance y is not only normally distributed about x, but has same variance (Fig. 13.3).

Again, as with ANOVA, we will examine these assumptions by looking at the residuals or error terms though histograms, normal plots, and plots or residuals against the predictor variables.

[3] If this terminology and these acronyms appear unfamiliar, refer back to that description.

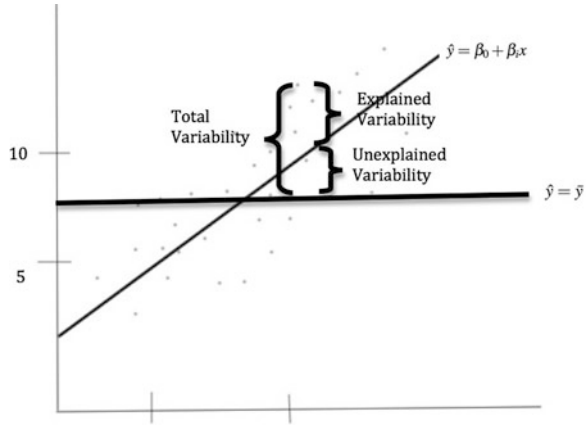


Fig. 13.2 Partitioning the variance

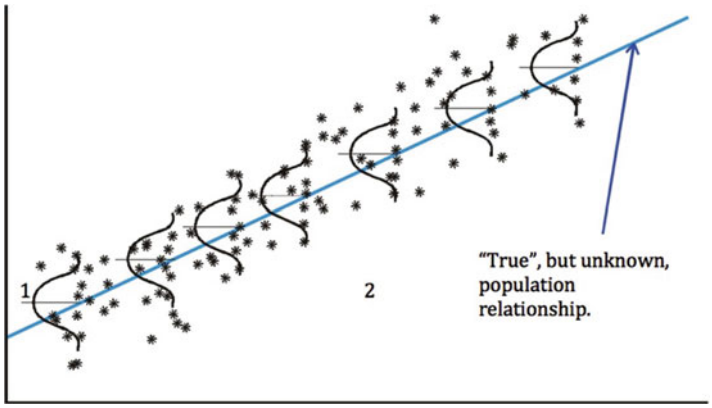


Fig. 13.3 Homoscedasticity (Image courtesy of SAS Institute)

13.2 PROC REG

As we noted in our discussion of ANOVA, PROC GLM provides the tools to analyze a regression model. We will use PROC REG, though because it is convenient (and easy to remember) and comes with a rich set of tools, particularly with respect to residual analyses and diagnostics.

At its most basic, PROC REG consists of a call to the procedure, specifying the data set, and a model statement.

```
Proc reg data=your.data;
          Model continuous_outcome_variable =
            continuous_predictor_variable;
          Title simple linear regression;
Run;
Quit; /* need to quit out of the procedure */
```

13.2.1 Regression Results

As with PROC GLM, PROC REG returns an ANOVA table. You will recall, that the Model Sum of Square represents the explained (or the ANOVA between groups) variance, and the error sum of squares represents the unexplained (or the ANOVA within group) variance. The F-test is the ratio of the explained (model) variance / unexplained (error) variance. The p-value associated with the F statistic tests the null hypothesis that $\beta_1 = 0$, or that our predictor (x) is not statistically significantly related to our outcome (y).

Look first at this p-value for the overall model. If it is statistically significant, then look at the individual t-test(s) for your predictor variable(s) and the R^2 (coefficient of determination) value to see how much of the variability your model explains.

You will also find results for the root mean square error (Root MSE) which is an estimate of the standard deviation of the response variable at each value of the predictor variable. It is (logically enough) the square root of the MSE.

The coefficient of variation (Coeff Var) is (as before) the size of the standard deviation relative to the mean. As in ANOVA it is calculated as[4]

$$\frac{\text{Root MSE}}{\bar{y}} * 100. \tag{13.4}$$

The coefficient of variation is a measure of data spread, as it is in univariate statistics. It is unit-less, so can be used to compare data that has different units of measurement or different magnitudes of measurement.

THE RETURN OF R

Note that the square root of the R^2 from a simple linear regression is in fact the Pearsons correlation coefficient (r) we met in the previous chapter on correlation.

You will also find, in your SAS regression results, a statistic called "Adj R Sq." This is the adjusted R^2 or the coefficient of determination that is adjusted for the number of predictor variables in your model. For simple linear regression with one predictor variable, it is not very useful, but as we will see, it becomes increasingly useful as the number of terms in your model increase.

[4]You may be beginning to appreciate why an understanding of ANOVA puts us in good stead for an understanding of regression. And, perhaps, that there are common themes that run throughout statistics, particularly when working with members of the general linear family.

13.2.2 Predicted Values

One of the nice things about regression in general (and PROC REG in particular) is that it allows you to predict values of your outcome variable (y) given some value for *x* that may not have been represented in your data set. Once you have your regression model, it's a fairly simple matter of plugging the value in which you are interested into the model and calculating the outcome. SAS automates the process for any number of predictions. The process takes three steps:

1. Create a temporary data set with the values for which you want predictions.
2. Append those values to the data set you used to create the model
3. Calculate the values by specifying a / p option following the model statement

13.2.3 Confidence and Prediction Intervals

Your predictions based on your regression model will, of course, be subject to error. These errors come in two basic flavors

- Confidence interval for a mean. A 95 % confidence interval that contains the population mean of *y* for a particular value of *x*. Since it is based on means, you will find your prediction is narrowest or more precise near the center (or mean) of your data, where you have more data points and becomes less certain as you move away from mean values of *y* and *x*
- Prediction interval for a single observation. Here you are interested in establishing an inference on an explicit, single observation. 95 % of the time, the prediction interval will contain the new observation. Because there is less data to work with and there is more variability than with sample means, these prediction intervals are wider than confidence intervals and relatively uniform.

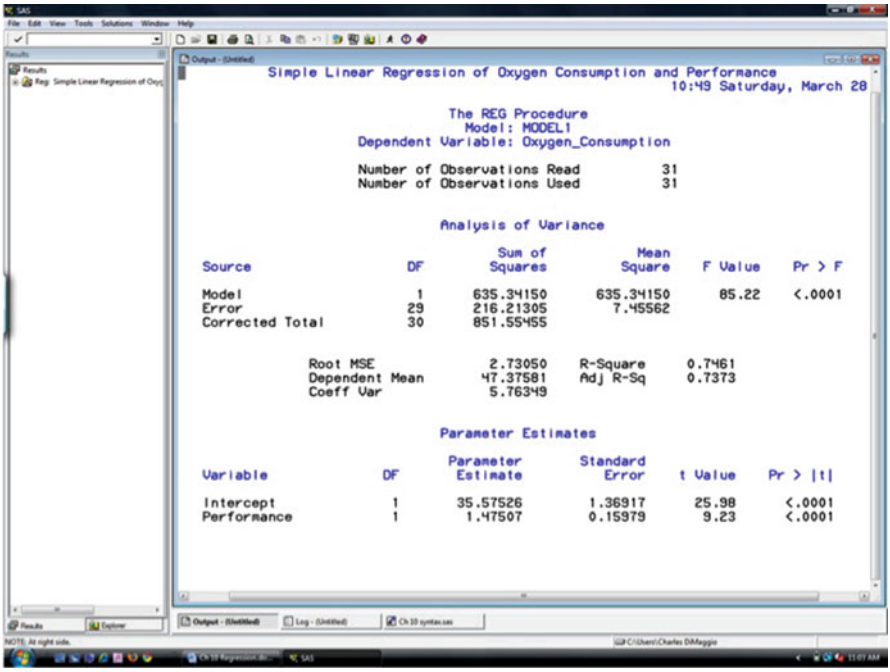
13.3 Demonstration of PROC REG

Let's return to the oxygen consumption data we've looked at before. We previously noted a linear relationship between performance and oxygen consumption. We will now explore this more closely with a data set that includes additional variables.

Our simple regression model in PROC REG is:

```
proc reg data=fitness;
   model oxygen_consumption=performance;
   title 'Linear Regression Oxygen Consumption and
     Performance';
run;
quit;
```

And returns the results (Fig. 13.4):



Simple Linear Regression of Oxygen Consumption and Performance
10:49 Saturday, March 28

The REG Procedure
Model: MODEL1
Dependent Variable: Oxygen_Consumption

Number of Observations Read 31
Number of Observations Used 31

Analysis of Variance

Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
Model	1	635.34150	635.34150	85.22	<.0001
Error	29	216.21305	7.45562		
Corrected Total	30	851.55455			

Root MSE	2.73050	R-Square	0.7461
Dependent Mean	47.37581	Adj R-Sq	0.7373
Coeff Var	5.76349		

Parameter Estimates

Variable	DF	Parameter Estimate	Standard Error	t Value	Pr > |t|
Intercept	1	35.57526	1.36917	25.98	<.0001
Performance	1	1.47507	0.15979	9.23	<.0001

Fig. 13.4 PROC REG

We see that the F-statistic for the model is statistically significant, that the t-test for our predictor variable performance is also significant, and that our model explains about 7 % of the variance in the data. Now let's look at our β regression coefficient values.

β_0 is our baseline or intercept term. It is, in general, the value you would expect for your outcome variable, *y*, if your predictor variable was zero. This is actually quite a helpful epidemiologic feature of linear regression, where we are very interested in unexposed or referent categories of individuals. If though our predictor variable cannot be reasonably expected to ever actually be zero (e.g. if our predictor was systolic blood pressure), then this baseline has no meaningful interpretation.

The value for β_1 is the rise over the run, or the change in our outcome variable for every 1 unit change in our predictor variable. Here, each 1 unit increase in performance results in about a 1.5 unit increase in oxygen consumption.

Even from this simple example, you can see how we gain much more information from a regression model than we do from a simple correlation.

RISE OVER RUN

With regard to the concept of "rise over the run," it is important to note that what might appear to be a small β value may have meaningful and important implications depending on the scale with which the predictor is measured. So, for example, if we measure age in months and our outcome is systolic blood pressure, a $\beta = 0.1$ would mean a greater than 1 mm Hg increase in blood pressure for every year of age. This quickly becomes clinically relevant.

Lets say we wanted to create a set of predictions based on this model. PROC REG has tools to make this fairly simple.

First, we create a data set of predictor (x) values for the performance variable.

```
data need_predictions;
  input performance @@;/* tell sas >1 value for same
    variable */
  datalines; /* tell SAS next line are data values */
0 3 6 9 12
; /* semi-colon on its own line*/
run;
```

Next, we append this data to the existing fitness data set

```
data predoxy; /* appending above data set to the fitness
  data set */
   set fitness
       need_predictions;
run;
```

Finally, we run the model using the values for which we want predictions

```
proc reg data=predoxy;
   model oxygen_consumption=performance / p; /*tells SAS to
                                              predict values
                                              for oxygen*/
   id performance;
   title 'Oxygen_Consumption=Performance with Predicted
     Values';
run;
quit;
```

We will find our predictions on the last five rows of our output (Fig. 13.5).

Another neat feature of PROC REG is that it easily allows to create and plot confidence and prediction intervals around our regression line. The following syntax accomplishes this. PROC REG recognizes the key words *clm* (confidence limit for mean), *cli* (confidence limit for individual), *conf* (to request overlaid plots of confidence intervals), and *pred* (to request overlaid plots of predicted values)

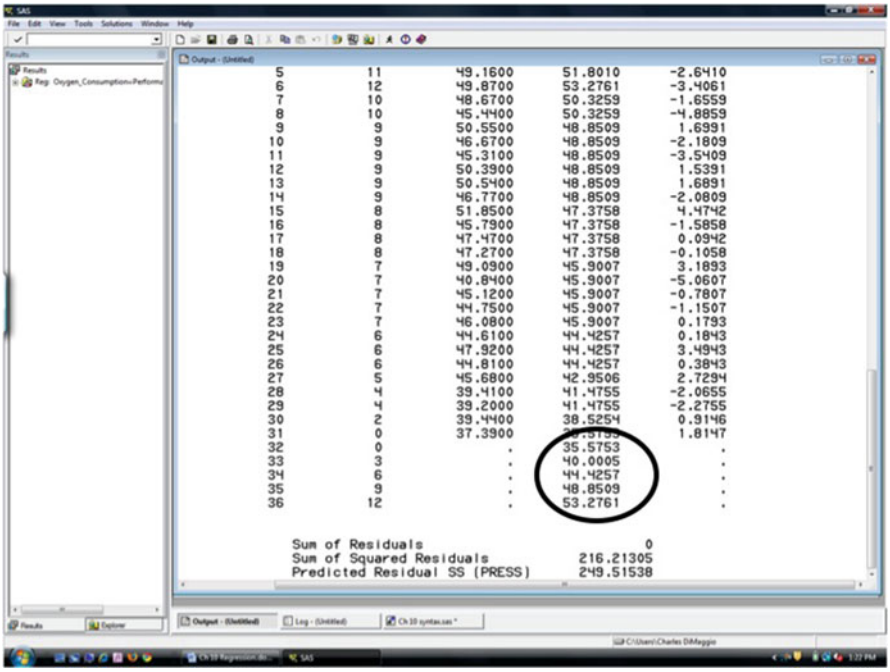


Fig. 13.5 Predictions from PROC REG model

options ps=50 ls=76;

```
goptions reset=all fontres=presentation ftext=swissb
  htext=1.5;
proc reg data=predoxy; /*using data set you created for
  prediction*/
   model oxygen_consumption=performance / clm cli
     alpha=.05;
   id name performance;
   plot oxygen_consumption*performance / conf pred;
   symbol1 c=red v=dot;
   symbol2 c=red;
   symbol3 c=blue;
   symbol4 c=blue;
   symbol5 c=green;
   symbol6 c=green;
   title;
run;
quit;
```

We get the following plot in our output[5]:

The actual values for the confidence and prediction intervals can be found in the output window printout (Fig. 13.6).

[5]Notice how the confidence interval for the mean becomes less precise the farther we get from the center of our data.

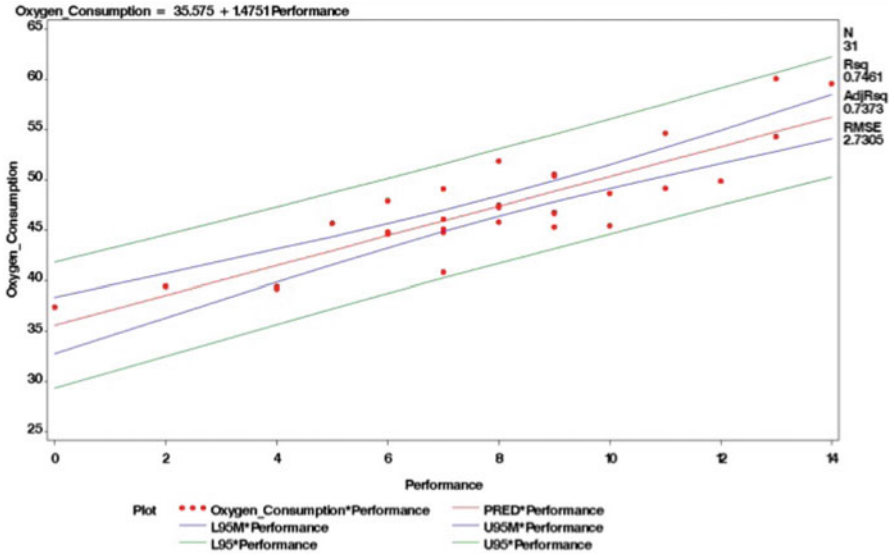


Fig. 13.6 Confidence and prediction intervals for a regression line

13.4 Multiple Regression with PROC REG

As you might imagine, in epidemiology we are much more likely to be interested in the relationship between several variables and some outcome, where the variables might be an exposure, confounders, mediators etc. When the outcome is continuous, multiple regression may be the statistical method of choice. But, our null hypothesis subtly but crucially changes. We now posit that our continuous response variable, y, does not change or is not related to more than one predictor variable, $x_1, x_2, \ldots, x_j$.

In simple linear regression, we could represent the model (simply) as a straight line representing the relationship between y and x. With multiple linear regression, our ability to represent the model is quickly outpaced by the complexity of the underlying geometry. For example, the relationship between two predictors, x_1 and x_2, could be represented by a plane.

Here we see the representation of our null hypothesis for two predictors (Fig. 13.7).

Here we see a relationship between our predictor variables and the outcome (Fig. 13.8).

After 2 predictors, the picture becomes complex enough that it can't be easily represented or conceptualized. Fortunately, the basic modeling ideas remain pretty much the same as they did with simple linear regression.

You will recall that in n-way ANOVA, our null hypothesis was that *all* categorical predictors were not related to the continuous outcome variable. This is also the case in multiple linear regression, where our null hypothesis is that all the regression coefficients (or all the slopes) are zero.

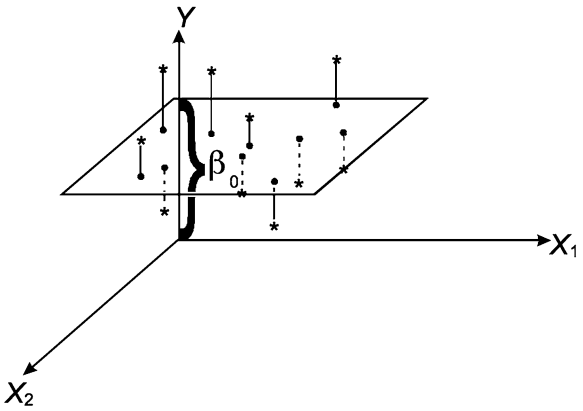


Fig. 13.7 Null hypothesis for two-variable linear regression (Image Courtesy of SAS Institute)

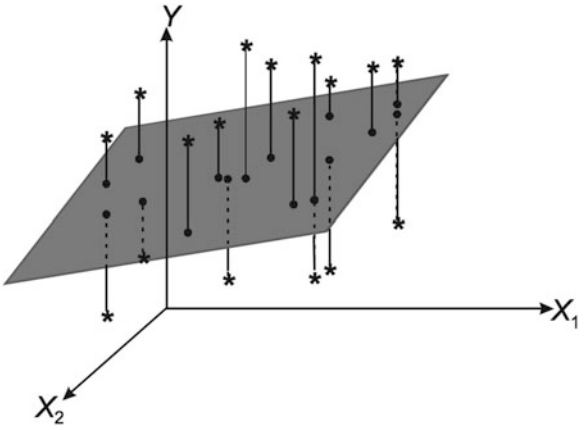


Fig. 13.8 Association for two-variable linear regression (Image Courtesy of SAS Institute)

Our assumptions are, again, that our observations are independent, that there is a linear relationship between predictors and outcome, and that error is random, normally distributed, and homoscedastic. Again, we will examine the error terms as mirrors of the *y* variable. The advantages of this approach becomes clear, because beyond a minimal number of variables we quickly outstrip our ability to check replicate values of *y* for every possible combination of predictor variable values.

As noted, the major advantage of multiple linear regression over simple linear regression is it may better reflect the complex interplay of multiple variables seen in the real world. Epidemiological processes invariably depend on more than one variable, and multiple regression may allow you to investigate these processes. Perhaps the main disadvantage is that this increased complexity makes it difficult to determine which model is best. This is something we will discuss in the next chapter.

Multiple regression has been traditionally divided into two main types, based on the use to which it is applied. In predictive modeling we are interested in creating a model into which we can plug in values of predictor variables so as to determine the

most likely outcome or response value. In explanatory modeling, we are interested in examining, testing, and describing the role of the relationship between an estimate and the outcome. This latter type is pretty much where we live in epidemiological applications of multiple linear regression. Usually (though not always, as we will see in our discussion of model selection) a model that is good for prediction is also good for explanation, so the distinction is in some respects artificial.

13.5 Interpreting Coefficients

The SAS syntax involved in multiple linear regression is very like that of simple linear regression, but the interpretation is a more involved. Perhaps the most important and powerful distinction from simple linear regression is that each β_i coefficient is interpreted as the effect or relationship between that predictor and the outcome, holding all the other predictor variables constant. This is the adjustment or control to which one refers when we state that the results of our multiple regression analysis adjusted for or controlled for other, potentially confounding variables.

As in simple linear regression, the intercept term is our baseline, or the value of our outcome if all the predictors or slopes were zero. Again, this is meaningful only if our predictors can be expected to ever be zero. If not, the intercept simply represents the point at which the regression line crosses the y axis.

Continuous predictor variables are similarly interpreted as they were in simple linear regression as the rise over the run, or the change in our outcome variable for every one unit change in our predictor. But now, this interpretation is based on the assumption that the other predictor variables are being held constant. So, for example, the effect of a one unit increase in drug dose is valid for (say) all individuals of the same age.

13.5.1 Categorical Predictor Variables

We may at this point consider the interpretation of categorical predictor variables in the setting of linear regression. With categorical variables, we need a reference category against which to measure the effect of a change in the categorical variable. In the simplest case, say we have a two-level categorical predictor variable like gender. We will code one possible outcome (say male) as zero and the other (female) as one. Our regression coefficient then is the effect on the outcome or y variable of one gender vs. the other, or the average difference in y between the category for which x is zero (male or the reference group) and the category for which x is one (female or the comparison group).

A *dummy variable* is categorical variable with a referent class that can be included in a regression model.[6] A dummy variable is in actuality a set of dichotomous indicator variables that divide some categorical outcome into all possible levels. If the categorical variable has p possible outcome levels, we need to create $p - 1$ dummy variables. So, for example, say we have some 3-level age variable. We would create two dummy variables, each coded 0 and 1. The referent category is when both dummy variables are coded 0. As described above, each coefficient is the difference between that variable and the reference category.

Unfortunately, PROC REG (or PROC GLM for that matter) does not provide the kind of simple tools found in say PROC LOGISTIC or PROC GENMOD to create dummy variables for categorical variables in regression models. We have to code them by hand instead.

The first time folks encounter dummy variable coding, they may (ahem…) feel a bit like dummies themselves. The topic may be best understood through demonstration. We'll spend a little time looking at a couple of examples. Lets begin with a modified version of the fitness data set. Say we have a three level age category variable called age_cat, coded 0 if a participant is less than 30 years old, 1 if 30 to 40 years old and 2 if older than 40.

Since we have 3 levels in our categorical variable, we need to create 2 dummy variables, here called "dummy1" and "dummy2." If the age category variable is 1, the dummy1 variable is set to 1, otherwise dummy1 is set to 0. If the age category variable is 2, then the dummy2 variable is set to 1, otherwise the dummy2 variable is set to 0. By exclusion then, if the age category variable is neither 1 nor 2, both dummy1 and dummy2 are set to 0:

```
data fitness2;
set fitness2;
if age_cat=1 then dummy1=1;
else dummy1=0;
if age_cat=2 then dummy2=1;
else dummy2=0;
run;
```

We can then use the following syntax to test the dummy variables.

```
proc reg data=fitness2;
   model oxygen_consumption=performance dummy1 dummy2;
   agedum: test dummy1, dummy2;
run;
quit;
```

Let's further explore dummy variable coding with another example. I'll just present the code here, and expect you to run it yourself to follow the results.[7]

[6] Remember, since we are dealing with *linear* regression, the *outcome* is still continuous.

[7] From Introduction to SAS. UCLA: Academic Technology Services, Statistical Consulting Group. (accessed April 7, 2009).

We begin by creating a data set. Let's say they are a set of scores on some test.

```
DATA dummy;
  INPUT id group score;
CARDS;
1 1 48
2 1 49
3 1 50
4 2 17
5 2 20
6 2 23
7 3 28
8 3 30
9 3 32
;
RUN;
```

We calculate the overall score and then the score for each subgroup. We see that the group means differ from one another:

```
PROC MEANS DATA=dummy; /* overall mean */
  VAR score;
RUN;

PROC MEANS DATA=dummy; /* mean for each group: group means
                          different */
  CLASS group;
  VAR score;
RUN;
```

We run a standard ANOVA and satisfy ourselves that group status is indeed important.

```
PROC GLM DATA=dummy; /* run standard ANOVA: group is
                        important */
  CLASS group ;
  MODEL score = group ;
RUN;
```

Let's create a dummy variable for group status and run a linear regression on score with group status as the explanatory variable:

```
DATA dummy2; /* create dummy variables using 3 indicators
                for group */
  SET dummy;
  IF (group = 1) THEN group1 = 1; ELSE group1 = 0;
  IF (group = 2) THEN group2 = 1; ELSE group2 = 0;
  IF (group = 3) THEN group3 = 1; ELSE group3 = 0;
RUN;

PROC REG DATA=dummy2; /* run proc reg using 2-level dummy
                         variables */
  MODEL score = group1 group2 ;
RUN;
```

If you run this code, you will notice a couple of things. First, the results are identical to those we got when we ran the ANOVA. This should, by now, come as no surprise. As we have seen, they are both based on the general linear model we have been talking about for a few chapters. The second thing you may notice is that we only used two of the indicator variables we created in the previous step. Why? Because by process of elimination, the un-included variable is the referent value. In the setting of linear regression, it is the intercept or the point at which the regression line crosses the *y* axis when the other variables are zero.

We see that the parameter estimate for the intercept (30) is the same as the mean for group 3 from PROC MEANS. The other parameter estimates are similarly related to the mean of the referent category. The parameter estimate for group 1 is the mean of group1 minus the mean of group 3 $(49 - 30 = 19)$, and the parameter estimate for group 2 is the mean of group 2 minus the mean of group 3 $(20 - 30 = -10)$. In summary,

```
Ypred for group1 = 30 + 1 * 19 + 0 * -10 = 49
Ypred for group2 = 30 + 0 * 19 + 1 * -10 = 20
Ypred for group3 = 30 + 0 * 19 + 0 * -10 = 30
```

SAS has more automated ways to do dummy coding in other PROCS (e.g., LOGISTIC, GENMOD) where you essentially just put in coefficients for comparisons. If we were coding these three groups in one of the PROCS, it would look something like this:

```
Ypred for group1: 1  0
Ypred for group2: 0  1
Ypred for group3: 0  1
```

13.5.2 Demonstration of Multiple Linear Regression

Linear regression is inherently an additive model. Each coefficient is the additive effect of that variable on the outcome. It is not, in actuality, the *isolated* effect of that variable on the outcome. Rather, it is the *additional* effect of that variable, *given* all the other variables in the model. This has important implications for model building and explanation. First, adding or removing a variable to a model will change the coefficients for all the other variables remaining in the model. Second, variables are often associated with each other on some level, which may cause additional complex changes in the model when variables are added or removed. The *beta* coefficients in a model will change depending on the variables we include. Clearly, our choice of variables is a critical consideration. We will address this issue in some detail in the next chapter.

But first, we return to our fitness example. Oxygen consumption is (again) our continuous outcome variable. We already know that performance is an important predictor variable. We will now include the additional predictor variable runtime:

```
/* multiple reg demonstration fitness data */

proc reg data=fitness;
   model oxygen_consumption=performance runtime;
   title 'Multiple Linear Regression fitness Data';
run;
quit;
```

We will see that, again, the overall model is statistically significant. But look at our parameter estimates.

Variable	DF	Parameter Estimate	Standard Error	t Value	Pr > |t|
Intercept	1	55.37940	33.79380	1.64	0.1125
Performance	1	0.85780	1.06475	0.81	0.4272
Runtime	1	-1.40429	2.39427	-0.59	0.5622

Given that the overall model is significant, we would certainly expect at least one of the predictors to be significant. But something has gone wrong. Here it is likely to be multicollinearity. The two predictors are so closely intertwined the regression methodology can't separate their effects. We will take this as a convenient point to stop and move onto the next chapter where we consider regression diagnostics, which address the potential problems of collinearity, influential observations, heteroscedasticity and non-independent or correlated errors.

Problems

13.1. Infant Birthweight and Hospital Charges: Assumptions
Lets examine the relationship between infant birthweight and overall inpatient charges using the infants data set we created. Begin by examining the normality assumptions of the two variables (Hint: Run a PROC CONTENTS to remind yourself of the variables).

What are your conclusions about the assumptions?

13.2. Infant Birth Weight and Hospital Charges: Correlation
Now, examine the correlation between these two variables using PROC CORR.

Is there a significant relationship between birth weight and charges? How would you describe this relationship?

13.3. Infant Birth Weight and Hospital Charges: Linear Regression
Write and run a PROC REG model testing the predictive value of infant birth weight on total charges. Exclude infants with birth weight equal to zero.

What is result for the overall F-test? Is birthweight a statistically significant predictor of total charges? How much of the total variability does the model explain? What is the relationship between the Pearsons correlation coefficient and your estimate of total explained variance?

Chapter 14
Regression Diagnostics

Abstract In this chapter, we consider how we may determine if there are problems with our underlying assumptions for the use of linear regression. We learn that residuals are the key to regression diagnostics, that SAS provides many tools, from plots to statistics, that help us examine whether our data meet assumptions such as normal distribution, linear relationships, and homoscedasticity, and if there are outliers influencing summary statistics.

14.1 Introduction

At the end of the previous chapter, we examined the effect of two predictor variables ("performance" and "run time") on a continuous outcome ("oxygen consumption"). We found that while the overall model was statistically significant, neither of the predictors were significant. One predictor that had been significant in a simple linear regression fell out of significance when we added the second variable. I mentioned that there was possibly a problem with collinearity between the two variables.

In this chapter we explore this possibility by examining if there are problems with our underlying assumptions for the use of linear regression.[1] The first key step in regression diagnostics is residual analysis, and we will start there.

As we first noted in the chapter on ANOVA, residuals are the key to diagnostics because as the only random variable on the right side of the regression equation, they mirror the random outcome variable on the left side of the equation. The error terms allow us to check our assumptions because we are unable to replicate values of our outcome variable for every possible combination of predictor variable values.

[1]Regression diagnostics should, really, come before the actual analyses. But, it's a bit of a "Catch 22" situation where you need to understand something about regression before the diagnostics make sense.

C. DiMaggio, *SAS for Epidemiologists: Applications and Methods*,
DOI 10.1007/978-1-4614-4854-9_14,

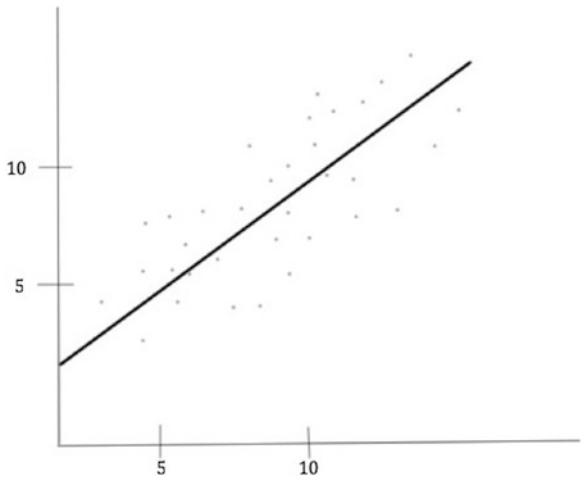


Fig. 14.1 The regression line

There are a number of summary statistics related to residual terms, but we will begin by examining plots. The following four plots[2] illustrate the importance of examining our data rather than relying on summary statistics.

Consider first this model estimated as $y = 3.0 + 0.5x$ with $R = 0.67$ (Fig. 14.1).

Under most circumstances, we would consider this to be a fairly good model. Troublingly, the data points for the following three plots return the exact same model with the exact same R value (Fig. 14.2).

SAS provides a rich set of tools to help us uncover this and other potential problems in our data that might lead us to wrong conclusions.

14.2 Residuals Redux

Residual analysis helps us uncover problems with our assumptions. Recall that residuals are the difference between our predicted values based on our model, and the actual data points. They represent the error or unexplained variance in our model (Fig. 14.3).

Similary recall that our assumptions for linear regression include independent observations, a normally distributed outcome variable at each value of the predictor variable, the same variance for our outcome variable at each value of the predictor variable and normally distributed error terms with a mean of zero.

[2]These four data sets were constructed by Francis Anscombe in the 1970's and are collectively referred to as "Anscombe's Quartet."

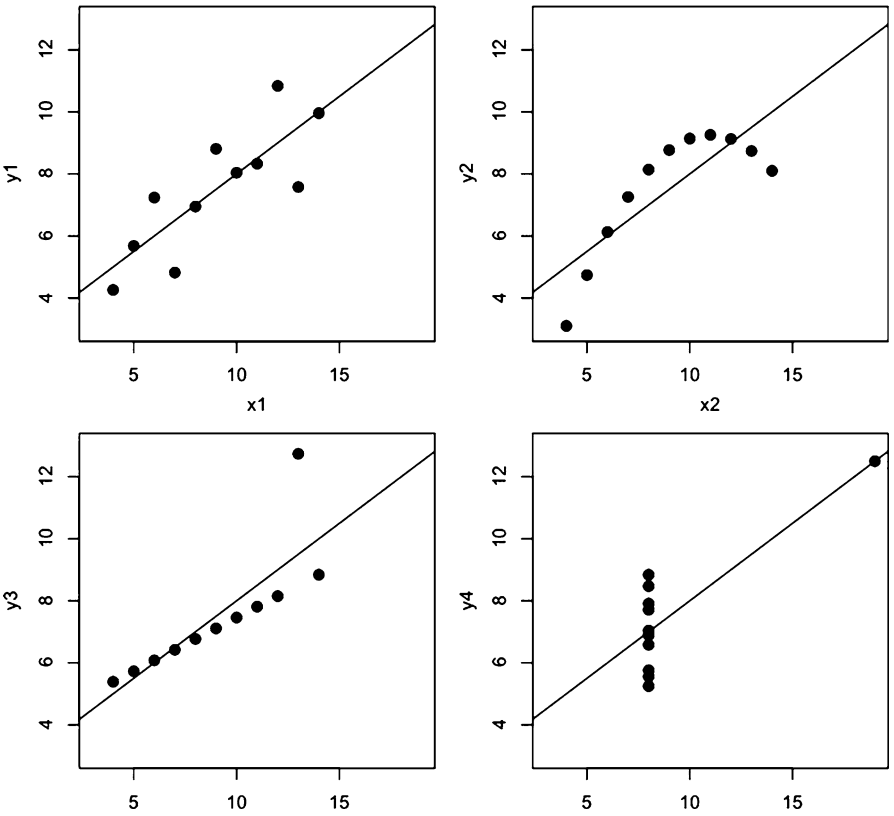
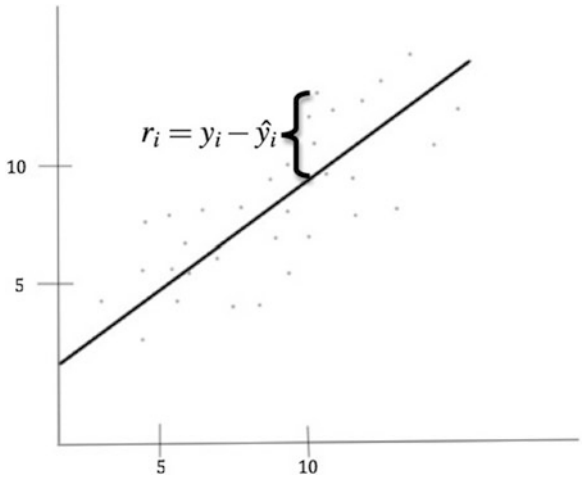


Fig. 14.2 Anscombe's Quartet

Fig. 14.3 A residual



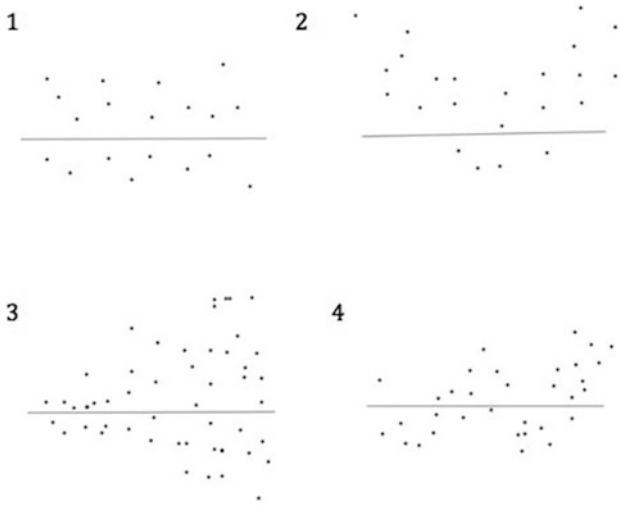


Fig. 14.4 Residual patterns

14.2.1 Residual Plots

As we did in ANOVA, we will plot our residuals against our predicted values. We will now, though, see a scatter or grouping of points. Ideally, we will see a random scatter of points around a reference line of zero. A U-shaped pattern indicates nonlinearity. A funnel-shaped pattern indicates heteroscedasticity. A sine-shaped pattern indicates nonlinearity, a violation of normality, nonindependence or perhaps a time series effect[3] (Fig. 14.4).

In ANOVA, we first created a set of residuals and fed them into PROC GPLOT and PROC UNIVARIATE. We will see that PROC REG has some easier residual tools. We'll first plot our residuals against our predicted values. If we see no problem on the residual by predictor plot, we can usually stop there. If we see a problem, we will look at the individual predictors to see where the violations of our assumptions may have come from, what type of violation may be occurring, and how we may address the violation, for example, with a transformation.

Keep in mind that linear regression is, in fact, a pretty robust procedure[4] especially when there are the kinds of numbers we often see in the secondary analysis of epidemiologic data. We tend to be concerned with gross violations of our assumptions.

[3] If, in fact, you are interested in looking for time series patterns, residual analyses may be helpful.

[4] Although independence is a *sine qua non.*

14.3 Outliers (Influential Observations)

Outliers are those values that are grossly out of line with the rest of the data. How large a value needs to be to be of concern is not always an easy question. And what to do about outliers is even thornier. We can test for outliers with so-called studentized residuals. These are error terms or residuals that are divided by their standard error:

$$\frac{\varepsilon}{s.e.(\varepsilon)}. \tag{14.1}$$

The rule of thumb is that a studentized residual with an absolute value less than two is likely to have occurred by chance. Studentized residuals between 2 and 3 are infrequent, occurring less the 5 % of the time, and may be outliers (the smaller the sample size, the more the concern). A studentized residual greater than 3 rarely occurs by chance alone and should be investigated.

Two other statistics can help in our search for influential or extreme observations which can cause the shape of the regression line to be different than it should be:

- Cooks D statistic measures the change in the parameter estimates that results from deleting each observation. The suggested cut off is $D > \frac{4}{n}$, where n is the sample size.
- The jackknife residual is based on the studentized residuals with the underlying concept that the very presence of an outlier affects the prediction line itself. The jackknife residual removes the outlier and plots the difference between the prediction line with the outlier and that without the outlier.[5] A Jackknife residual greater than $|3|$ is considered potentially problematic (Fig. 14.5).

If we determine that one of our data values is indeed an outlier, the question then becomes what to do about it. We don't want to just throw out the value. The first approach is to check the data and correct what may possibly be an error. You may have to collect more data. You may have to consider a different model. Sometimes higher-order terms like squares can help.

14.4 Collinearity

Collinearity refers to when two or more explanatory variables are highly correlated. Typically, we'll see a statistically significant test for the overall model, but the individual predictors are either not significant or have strange, or apparently incorrect, slopes (e.g., with the wrong sign). It can be helpful to think of collinearity in terms of Venn diagrams. Here we see that the variable X_1 accounts for about 25 %

[5] One can imagine the prediction line jackknifing up and down, hence the name.

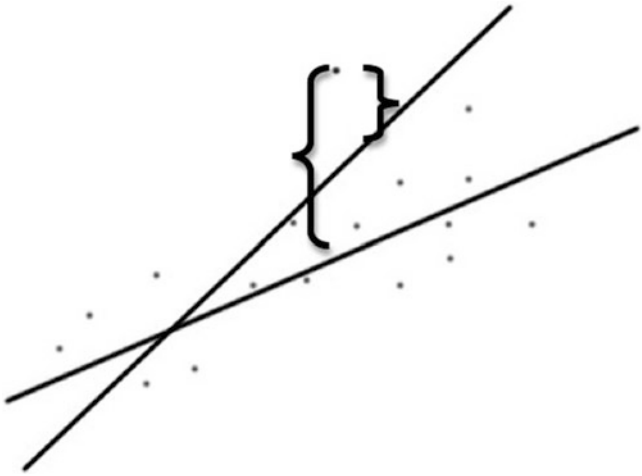


Fig. 14.5 Jackknife residual

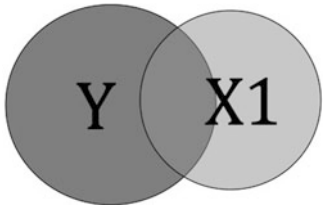


Fig. 14.6 Variance in Y explained by X_1

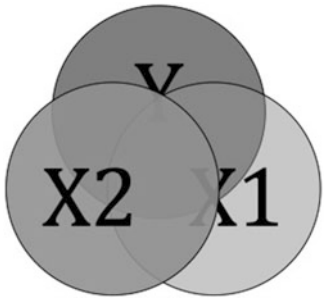


Fig. 14.7 Variance in Y explained by X_1 and X_2

of the variability in the outcome Y (Fig. 14.6). The model R^2 would be 0.25. Let's assume that the model itself $Y = \beta_0 + \beta_1 X$ is statistically significant at $p = 0.001$.

Now imagine introducing another variable, X_2, which you believe is also associated with Y. If, though, X_1 and X_2 are also associated with or correlated with each other, each of their individual effects will appear smaller (Fig. 14.7). Because we are explaining more of the overall variance of Y, we may see that the overall model R^2 increases (say from 0.25 to 0.40), but because X_1 and X_2 are correlated, the correlation between X_1 and Y controlling for X_2 may decrease, say to 0.01.

If we were to add yet another correlated predictor variable (X_3), the correlation between X_1 and Y controlling for X_2 and X_3 will decrease even further, perhaps to insignificance. We can see (hopefully intuitively) that there is simply too much competition for that patch of variability in Y which X_1 had all to itself in our first model.

In a perfect world, we would like our predictor variables to be completely uncorrelated. Of course, such a perfect world does not exist, so we can expect some level of correlation among our predictor variables. As we will see, deciding which variables to keep in a model and which to exclude is really an epidemiological question, not a statistical one.

SAS comes equipped with a number of collinearity diagnostics. The most commonly used is probably the variance inflation factor (VIF), which is a relative measure of the increase in variance due to the collinearity of a variable with other variables.

$$VIF_i = \frac{1}{(1 - R_i^2)}, \tag{14.2}$$

where, for the variable X_i, R_i^2 is the R^2 value from a regression with X_i (normally a predictor variable) set as the outcome or dependent variable. So, for example, if the model is $Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3 + \beta_4 X_4$ we calculate R_3^2 for X_3 by fitting the model $X_3 = \beta_0 + \beta_1 X_1 + \beta_2 X_2$.

The underlying idea is actually quite simple. The higher the R_i^2 for a variable (and hence the more variance it explains among its co-predictor variables), the smaller the denominator and the larger the VIF. A $VIF_i > 10$ indicates collinearity. A drawback of the VIF is that while it provides a measure of collinearity, it doesn't tell which variables are collinear with each other.

Two additional SAS collinearity diagnostics are based on Eigen values. COLLIN and COLLINOINT have the advantage of indicating which groups of variables may be collinear. COLLIN includes the intercept, COLLINOINT, is adjusted for the intercept. The drawback of *these* statistics is that there are as yet no well accepted guidelines and thresholds for their use.[6]

14.5 Demonstration: Residual Diagnostics for the Fitness Data

Based on our discussion above, we may have an issue of collinearity in our oxygen consumption model. We begin our diagnostics with residual plots.[7]

[6]This according to SAS themselves. I invariably rely on VIFs and seek out guidance from my biostatistical colleagues for additional collinearity statistics.

[7]Being good statistical citizens, we would have conducted these regression diagnostics before conducting the actual analyses.

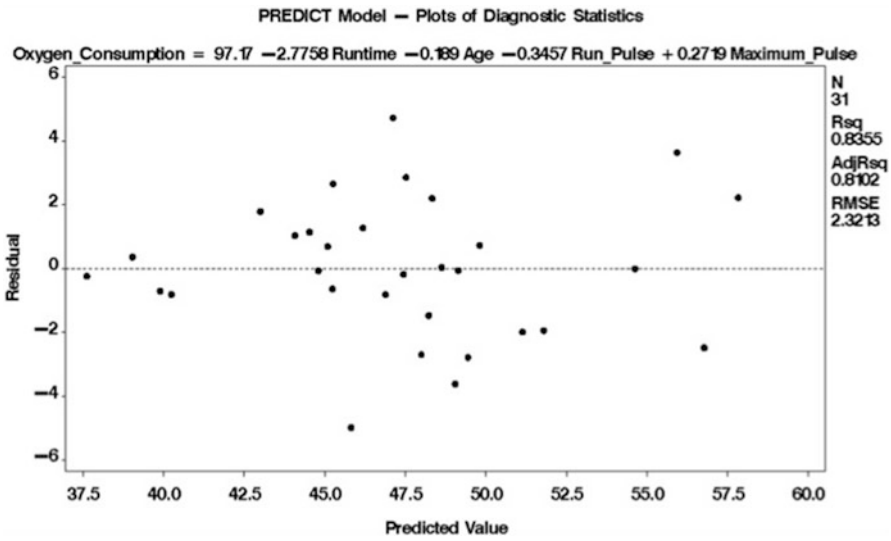


Fig. 14.8 Residual plot for the model

```
options ps=50 ls=97;
goptions reset=all fontres=presentation ftext=swissb htext=1.5;

proc reg data=fitness;
PREDICT: model oxygen_consumption
         = runtime age run_pulse maximum_pulse;
plot r.*(p. runtime age run_pulse maximum_pulse);
                    /*plot residuals v predicted values*/
plot student.*obs. / vref=3 2 -2 -3
                  /*studentized obs. Gives obs. # to ID */
                  haxis=0 to 32 by 1;
plot student.*nqq.; /*nqq another name for normal prob plot*/
symbol v=dot;
title 'PREDICT Model - Plots of Diagnostic Statistics';
run;
quit;
```

Our plot looks fine (Fig. 14.8). The data points appear randomly distributed on either side of the zero line. In practice, we could pretty much stop looking at residual plots here. Lets look at some of the individual predictor variables, though, so you can see that they are similar (Figs. 14.9 and 14.10).

Next, we begin our consideration of outliers with the request for a plot of studentized residuals vs. observation numbers (which we requested as part of the residual plots) (Fig. 14.11).

We see here that observations 15 and 20 have values beyond an absolute value of 2, indicating that they may be problematic. We now further explore possible influential observations. The following syntax requests an output consisting of

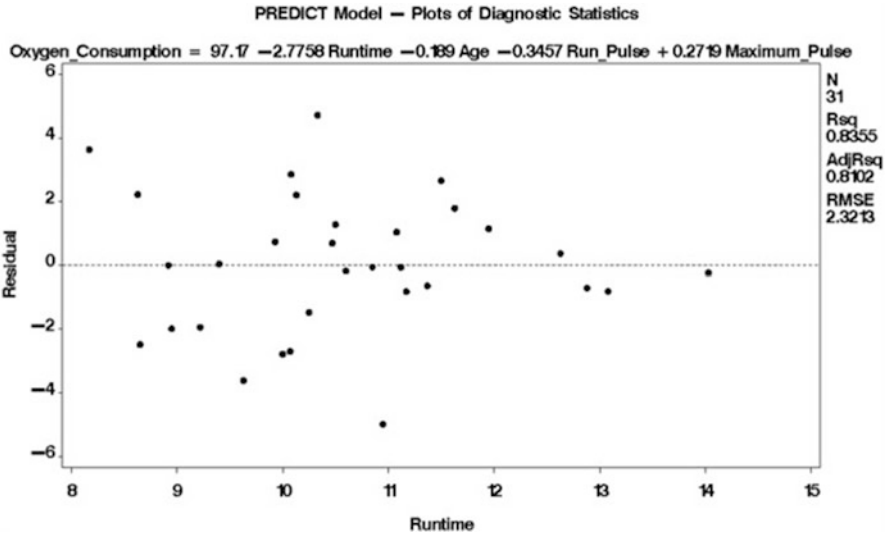


Fig. 14.9 Residual plot for the "Runtime" variable

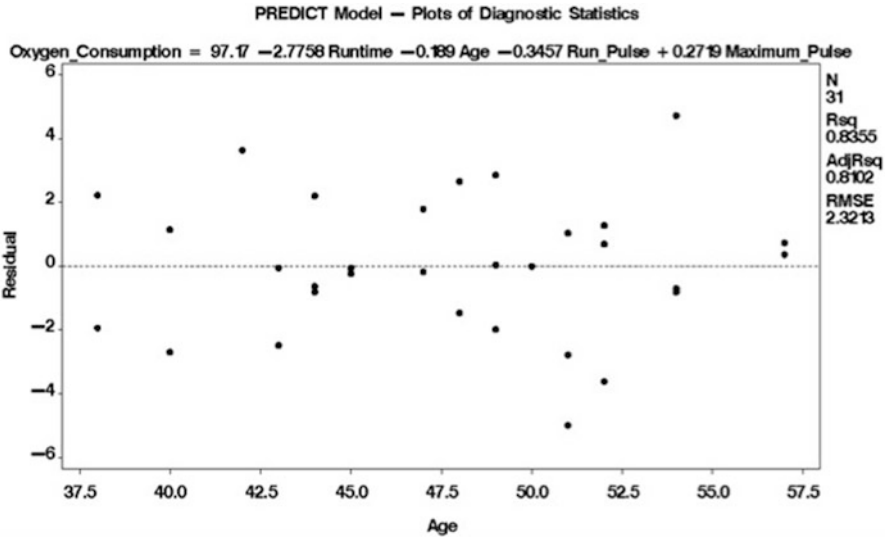


Fig. 14.10 Residual plot for the "Age" variable

residuals and so-called influence statistics. Note that we are also creating an output data set called ch4outliers in the work directory. We will use this to check for outliers.

```
goptions reset=all;
proc reg data=fitness;
```

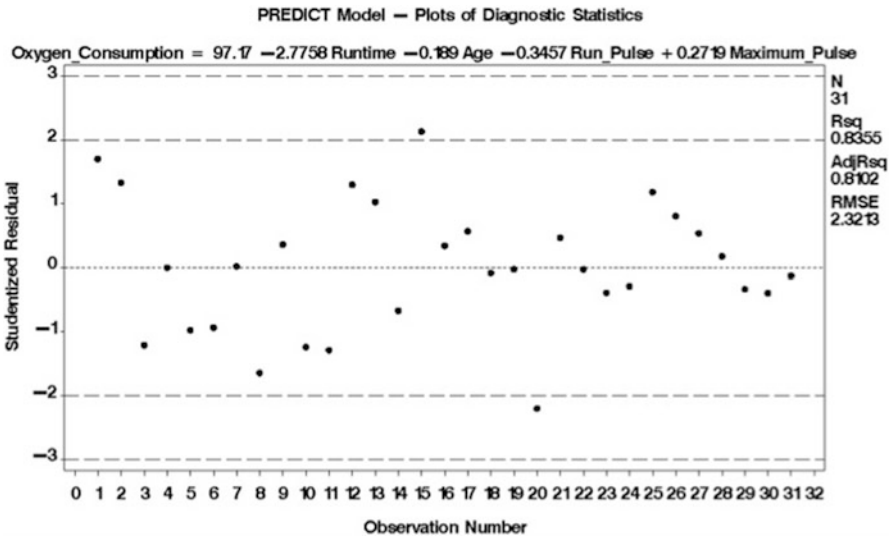


Fig. 14.11 Plot of studentized residuals

```
        PREDICT: model oxygen_consumption
                =runtime age run_pulse maximum_pulse
                / r influence;
/* r is residuals, influence is for the influence statistics*/
        id name; /* to allow us to identify the outlier*/
        output out=ck4outliers
               rstudent=rstud  cookd=cooksd;
    /* note we are creating an output data set of the outlier
    statistics so we can get SAS to look at them for us so we
    create and name a data set and label the output variables
    in which we are interested*/
    title;
    run;
    quit;
```

This data set is small enough that we could eyeball it to identify the studentized residual values for observations 15 and 20 that may be problematic. With all but the smallest data sets, though, we would want some way to perhaps automate the process. The following macro can be used to print out a list of influential observations:

MACRO

SAS Macros are small programs or routines that automate repetitive tasks. Some folks swear by them. I rarely, if ever, have had a reason to use them. But that might simply be resistance on my part to learning additional programming syntax.

(continued)

(continued)

A macro consists of special SAS syntax that contains unique language so it can be run repeatedly with different variables. In a SAS macro, "%let" creates a macro variable. A macro variable does not exist in any particular data set, but is available for use during in any procedure during a SAS session. The character "&" before a variable name tells SAS to retrieve that macro variable. You can just copy, paste, and adapt the following macro for use in your own analyses (That's what I do). If you are interested in learning more about SAS macros, this online document is a nice introduction.

```
/* MACRO FOR OUTLIERS */

/*  set the values of these macro variables, */
/*  based on your data and model.            */
%let numparms=5;  /* # of predictor variables + 1 */
%let numobs=31;   /* # of observations */
%let idvars=name; /* relevant identification variable(s) */

data influential;
   set ck4outliers;
   cutcookd=4/&numobs;

   rstud_i=(abs(rstud)>3);
   cookd_i=(cooksd>cutcookd);
   sum_i=rstud_i + cookd_i;
   if sum_i > 0;
run;

/* then print out the list of influential observations */

proc print data=influential;
   var sum_i &idvars cooksd rstud cutcookd
       cookd_i rstud_i;
   title 'Observations that Exceed Suggested Cutoffs';
run;
```

We see in the screen-shot that the macro prints out the observation for the participant "Gracie" as a possible outlier (Fig. 14.12). We may want to look at the original data to make sure that there are no errors in the observation for Gracie, or consider collecting additional observations on additional participants.

Our next step is to address the issue of potential collinearity in our data set. The following syntax requests a VIF:

```
proc reg data=fitness;
   FULLMODEL:
   model oxygen_consumption
                 = performance runtime age weight
                   run_pulse rest_pulse maximum_pulse
                 / vif ;/* collinearity diagnostic*/
```

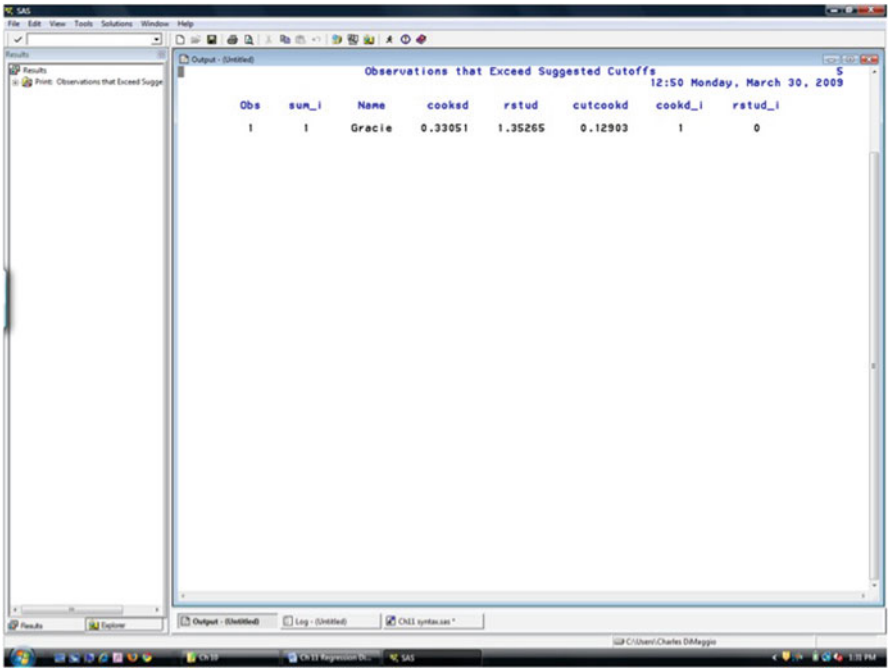


Fig. 14.12 Results of the outlier macro

```
   title 'Collinearity -- Full Model';
run;
quit;
```

We look for VIFs greater than 10 in the printout (Fig. 14.13).

The variables "performance" and "runtime" are clearly above the threshold. How best to deal with this collinearity? It might make sense to drop one of these two collinear predictors, but this decision must be informed by epidemiological and subject matter expertise.[8] In this case, let us suppose that performance and runtime are both essentially measuring the same thing and do not play different epidemiological roles in our model. Lets remove performance.

```
proc reg data=fitness;
   NOPERF:
   model oxygen_consumption
                 = runtime age weight
                   run_pulse rest_pulse maximum_pulse
                 / vif;
   title 'Collinearity -- Performance Removed';
run;
quit;
```

[8]This is one reason why regression diagnostics should be performed before model selection procedures.

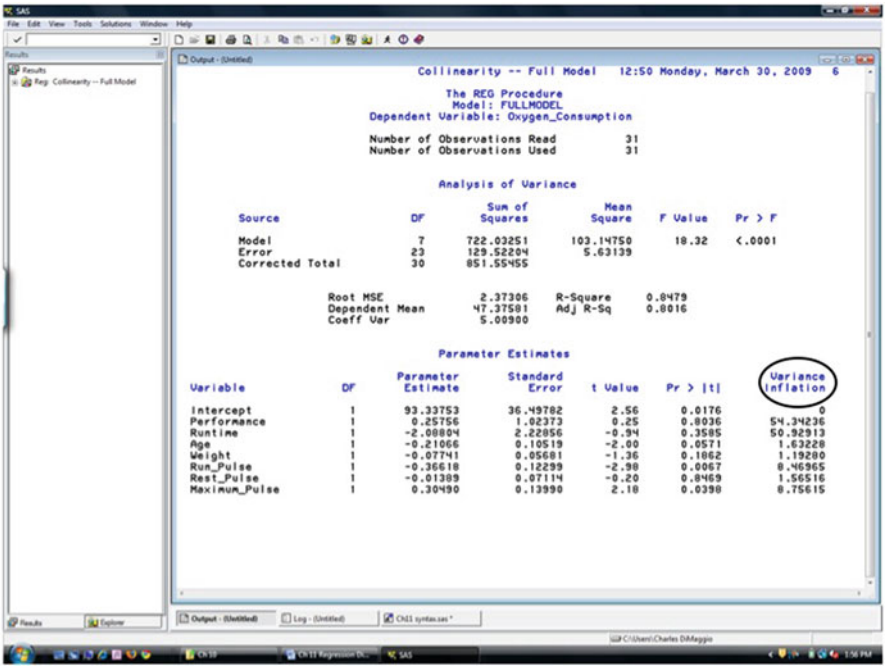


Fig. 14.13 Variance inflation factors

We now see that a number of our predictor variables are statistically significant, and none of the variables have a VIF above 10 (Fig. 14.14).

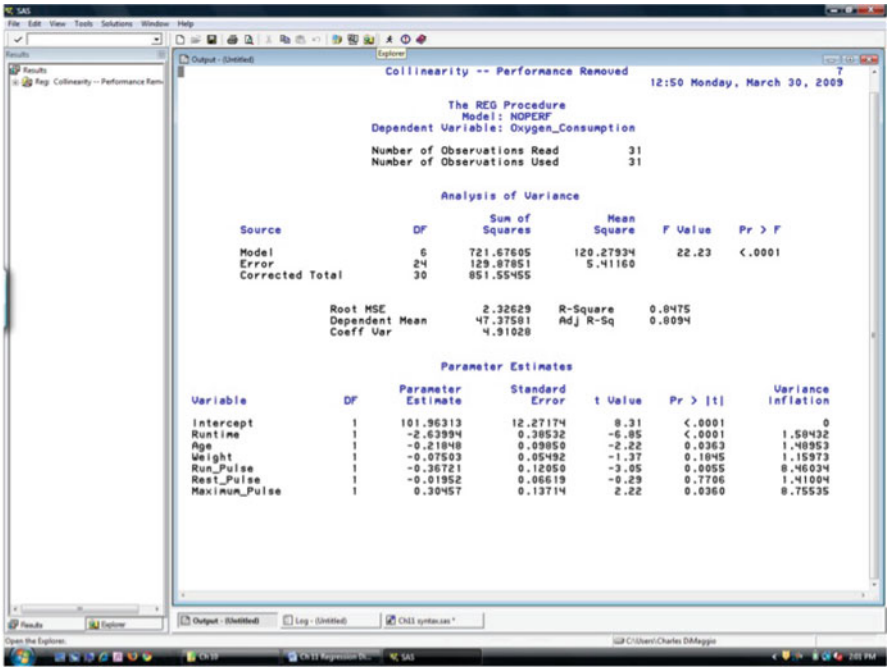
14.6 A Word About Model Selection

When folks refer to "model selection," they are most often talking about "variable selection", or deciding which variables to include in a regression. SAS has a number of tools available to ease the path to model selection. This is one of those cases, though, where the unpaved road may be preferable to the highway. I will present the SAS tools for automated model selection. Then I will try to explain why I don't use them.

14.6.1 SAS Model Selection Tools

The SAS linear regression model selection tool is, appropriately enough

```
/ SELECTION = <option>
```



Collinearity -- Performance Removed 7
12:50 Monday, March 30, 2009

The REG Procedure
Model: NOPERF
Dependent Variable: Oxygen_Consumption

Number of Observations Read 31
Number of Observations Used 31

Analysis of Variance

Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
Model	6	721.67605	120.27934	22.23	<.0001
Error	24	129.87851	5.41160		
Corrected Total	30	851.55455			

Root MSE	2.32629	R-Square	0.8475
Dependent Mean	47.37581	Adj R-Sq	0.8094
Coeff Var	4.91028		

Parameter Estimates

Variable	DF	Parameter Estimate	Standard Error	t Value	Pr > |t|	Variance Inflation
Intercept	1	101.96313	12.27174	8.31	<.0001	0
Runtime	1	-2.63994	0.38532	-6.85	<.0001	1.58432
Age	1	-0.21848	0.09850	-2.22	0.0363	1.48953
Weight	1	-0.07503	0.05492	-1.37	0.1845	1.15973
Run_Pulse	1	-0.36721	0.12050	-3.05	0.0055	8.46034
Rest_Pulse	1	-0.01952	0.06619	-0.29	0.7706	1.41004
Maximum_Pulse	1	0.30457	0.13714	2.22	0.0360	8.75535

Fig. 14.14 Variance inflation factors after removing a collinear variable

It is placed after the model statement as part of PROC REG. SAS makes the following selection options available to you:

- RSQUARE, ADJRSQ Runs all possible combinations of the variables and selects the "best" model based on the R^2 or adjusted R^2 statistic . So, if you specify k variables in your model, SAS will test 2^k models.
- FORWARD Runs all single predictor models, keeps all the variables that are statistically significant in the single variable scenario, then runs all 2-predictor models using those variables and keeps the variables that were statistically significant in the 2-predictor setting, then runs all the *3-predictor* models, etc
- BACKWARD Is (logically) the reverse approach to FORWARD. SAS runs an initial model with *all* the variables, keeps those that are statistically significant, then runs a model with *those* variables, retains the variables that are statistically significant, etc…
- STEPWISE Is a hybrid of FORWARD and BACKWARD where variables are included in a forward fashion, and excluded in a backward fashion[9]

[9]If this is starting to sound a little suspect, read on.

A syntax might look something like this:

```
Proc reg data=work.data;
Forward:  model outcome=pred1 pred2 pred3 / selection=forward
  slentry=0.001;
Backward:  model outcome=pred1 pred2 pred3 / selection=backward
  slentry=0.001;
Stepwise:  model outcome=pred1 pred2 pred3 / selection=stepwise
  slentry=0.001;
Title  ;
Run;
Quit;
```

Where the "slentry" option sets the level of statistical significance for which a variable is included or excluded from a model.[10]

14.6.2 Problems with Automated Selection Procedures

I am not a fan of automated model selection algorithms. They are convenient, and they are useful in some settings, but they represent an approach to data analysis and modeling that is not consistent with my goals as an epidemiologist. Let's take a look at some of the problematic aspects of automated model selection procedures.

1. They are based on p-values, and slavish devotion to p-values is itself the subject of considerable criticism.
2. Even if you are a fan of p-values, the SAS default p-values are too high (0.1–0.5), hence the need to set SLENTRY= (for forward selection) and SLSTAY= (for backward selection) to more reasonable levels.
3. Even if you set the entry and retain criteria to a more reasonable level, there is the issue of multiple significance testing (see the chapter on ANOVA).
4. Even if you adjust for multiple tests, you may (very likely) end up with a different model from each method.
5. Finally, and most importantly, automated selection methods do not take important causal thinking into account.

This last point is the deal breaker for me. As an epidemiologist I should have some idea about the role each variable in my model plays. Is it the exposure or risk factor that I believe is associated with the outcome? Is it a confounder that I may have to control for? Is it an interaction term? These are questions related to my knowledge of the subject, not something to be submitted to the black-box of p-values. Causal thinking drives analysis, not vice versa. Automated procedures are useful, I think, if your goal is to predict some outcome. Epidemiology (and science in general) aims to explain and intervene if possible.

[10] More on this in a moment.

14.6.3 Some Advice on Model Selection

What then is an epidemiologist to do? My usual advice is to base model selection as much as possible on your knowledge of the disease outcome in which you are interested. Look closely at your univariate associations. Stratify to assess confounding. Plot to assess interaction. Diagnose outliers and deviations from assumptions. Assess fit. Many of the things we've gone over in previous chapters.

If you must compare models based on statistical significance, then I suggest you do it explicitly rather than rely on automated procedures. SAS makes manual model comparison relatively painless through the use of so-called partial F-tests . Recall that an F statistic tests the null hypothesis that all the regression coefficients are zero and do not contribute to our explanation, understanding, or prediction of the outcome variable. A partial F-statistic tests whether the addition of an independent or explanatory variable, given others already in the model, significantly *contributes* to the prediction of the outcome variable.

So, for example, how does much the variable x_* contribute to a model given that the variables $x_1, x_2, \ldots, x_p$ are already in a model? We can look at it in terms of partitioning sums of squares:

$$SS_{x_*|x_1,x_2,\ldots,x_p} = SS_{\text{model:}x_1+x_2+\cdots+x_p+x_*} - SS_{\text{model:}x_1+x_2+\cdots+x_p}. \tag{14.3}$$

Where the sum of squares due to x_* is what remains when we subtract the sum of square of the model with x_* absent from that of the model wit x_* present. To test the statistical significance of the additional sum of squares, we calculate the F-statistic:

$$F_{x_*|x_1,x_2,\ldots,x_p} = \frac{SS_{x_*|x_1,x_2,\ldots,x_p}}{SS_{\text{model:}x_1+x_2+\cdots+x_p+x_*}}. \tag{14.4}$$

And, as usual, look up the associated p-value. Well, not really, as usual SAS does it for us. In fact, these partial F-tests are default statistics in PROC REG. Up until now, we've been ignoring them. They are the "Type I" and "Type II" sums of squares that SAS reports. To test a variable, use the (appropriately named) "TEST" option. The following syntax tests the joint significance of "c" and "d" given that "a" and "b" are in the model:

```
model y = a b c d / TEST c=0 d=0
```

For a single additional variable, they are essentially the same. If you are testing the addition of more than one variable, "Type I" tests the variables added in sequence, and "Type II" tests the variable added last, assuming all the other variables have been added.

Problems

14.1. Enter Data

The text file "grades.txt" contains data on a college entrance exam (first column) designed to predict college grade point average (second column). Read the data into a SAS file. Name the exam variable score. Name the "gpa" variable gpa.

14.2. Residuals

Create plots of the residuals by score and by the predicted values, and a plot of studentized residuals by observation number. Do the residual plots indicate any problems with the model assumptions? Are there any outliers? On what do you base your conclusion about outliers?

References

1. Cavalieri P, Marovich P, Patetta MJ, Walsh S, Bond C, SAS Institute (2000) Statistics I: Introduction to anova, regression, and logistic regression: course notes. SAS Institute, Cary, NC
2. Daniel WW (2006) Biostatistics: a foundation for analysis in the health sciences 8th edition with SPSS software CD Rom 14.0 set (Wiley series in probability and statistics). Wiley, New York
3. Darroch J (1997) Biologic synergism and parallelism. Am J Epidemiol 145(7): 661–668
4. Delwiche LD, Slaughter SJ (2008) The little SAS book: a primer. SAS Institute, Cary, NC
5. Hennekens CH, Buring JE, Mayrent SL (1987) Epidemiology in medicine. Lippincott Williamns & Wilkins, Philadelphia
6. Hosmer DW, Lemeshow S (2000) Applied logistic regression (Wiley series in probability and statistics). Wiley, New York
7. Kelsey JL, Whittemore AS, Evans AS, Thompson WD (1996) Methods in observational epidemiology. Oxford University Press, Oxford, New York
8. Kleinbaum DG, Kupper LL, Nizam A, Muller KE (2007) Applied regression analysis and multivariable methods (Duxbury applied). Duxbury Press, North Scituate, MA
9. Patetta MJ, Amrhein J (2005) Categorical data analysis using logistic regression: course notes. SAS Institute, Cary, NC
10. Rothman KJ, Greenland S, Lash TL (2008) Modern epidemiology, 3rd edn. Lippincott Williamns & Wilkins, Philadelphia
11. Schlesselman JJ (1982) Case-control studies: design, conduct, analysis. Oxford University Press, Oxford, New York
12. Susser ES, Schwartz S, Morabia A, Bromet E (2006) Psychiatric epidemiology: searching for the causes of mental disorders. Oxford University Press, Oxford, New York

C. DiMaggio, *SAS for Epidemiologists: Applications and Methods*,
DOI 10.1007/978-1-4614-4854-9,

Solutions

Chapter 2

2.1 Submitting Commands and Reading Output

```
/** compare ages upstate and nyc ami admissions with ttest **/

libname p8483 " ";

proc ttest data=p8483.demo$_1$; * identify the data
set you are working with;
class nyc; * identify the variable you are using to
identify and compare groups;
var age; /* continuous variable you are analyzing*/
title 'Comparing ages nyc and upstate ami patients';
*title for your output;
run;


/* univariate analysis of age */

proc univariate data=p8483.demo$_1$;
var age;
histogram; *returns histogram of age variable;
probplot;
run;
```

- The mean age of upstate AMI patients is 70.
- The mean age of NYC AMI patients 69.
- The difference is statistically significant. ($p < 0.0001$)
- It is not likely to be clinically important.

2.4 Using PROC CONTENTS To View Variables

- There are 502,492 observations in the data set.
- Each observation represents an acute myocardial infarction.

C. DiMaggio, *SAS for Epidemiologists: Applications and Methods*,
DOI 10.1007/978-1-4614-4854-9,

- There are 16 variables for each observation.
- The data is not sorted.
- The DATE variable is numeric.
- DISPO is a 2-unit long character variable.
- DATE looks like numbers.
- ECODE is missing from most observations, and is therefore not very useful.

Chapter 3

3.1 Reading in Data from the Editor Window

```
data test;
input
@1  ID     $6.
@7  RESULT $8.
@15 VALUE  3.;
cards;
203769Positive486
201948Positive400
202085Positive364
201755Positive416
202092Positive373
202087Positive657
201358Negative341
201429Positive448
201549Negative320
202741Positive391
201627Positive532
202004Negative268
202052Negative334
203531Positive573
204366Negative348
204042Negative252
;
run;

proc print data=test;
run;
```

3.2 Reading in Data from an External Source

```
DATA sparcs_1;
INFILE 'your/path/here/nycsparcs.TXT'  LRECl=452 OBS=100;
INPUT
@18   DATE yymmn6.
@44   AGE        3.
@50   COUNTY     $CHAR2.
@52   ZIP        $CHAR5.
```

```
@57   SEX        $CHAR1.
@58   RACE       $CHAR2.
@60   ETHNIC     1.
@71   PDX        $CHAR6.
@343  DISPO      $CHAR2;
run;

proc print data=sparcs_1;
run;
```

3.3 Creating a SAS Library

```
libname p8483 " C:\Users\Charles DiMaggio\Desktop";
```

Chapter 4

4.1 Using PROC PRINT

```
libname ex4_1;

proc contents data=;
run;

proc print data= (obs=20) noobs ;
var age pdx charge;
where los gt 14;
sum charge;
run;


proc print data=- (obs=20) noobs ;
var age pdx charge;
where los lt 7;
sum charge;
run;
```

- It appears to be hospital discharge data.
- there are 502,492 observations with 16 variables each.
- The total charges are $914,170.00.
- The total charges for patients whose length of stay was less than seven days is $104,660.00.
- Length of stay is associated with cost of care.

4.2 From SAS to Excel

```
ods html file =  style=minimal;

proc print data=- (obs=150) noobs;
var sex age zip date dispo pdx;
title 'Patient Demographics';
run;
```

ods html close;

4.3 Creating and applying formats

```
proc print data=       (obs=20) noobs;
var charge;
format charge dollar11.2;
run;

proc format;
  value dol_range low - 500000 = 'Low'
                  500001 - 1000000 = 'Medium'
                  1000001 - high = 'High';
run;
proc print data=p8483.demo_1 (obs=20) noobs;
var charge;
format charge dol_range.;
run;

proc print data=        (obs=20) noobs;
var sex;
run;

proc format;
  value $gender "F" = "Female"
                        "M" = "Male";
run;

proc print data=       (obs=20) noobs;
var sex;
format sex $gender.;
run;
```

4.4 Using Titles and Labels

```
data test;
input
@1  ID     $6.
@7  RESULT $8.
@15 VALUE  3.;
cards;
203769Positive486
```

```
201948Positive400
202085Positive364
201755Positive416
202092Positive373
202087Positive657
201358Negative341
201429Positive448
201549Negative320
202741Positive391
201627Positive532
202004Negative268
202052Negative334
203531Positive573
204366Negative348
204042Negative252
;

proc print data=test noobs label;

label
ID = 'Patient Identifier'
RESULT = 'Final Result'
VALUE = 'Assay Level';

run;

title 'Initial Blood Test Results';
run;
```

Chapter 5

5.1 Concatenating Data Sets

```
data tot_deaths; set tot_deaths; tot_911_deaths =
female_911_deaths + male_911_deaths; run;
```

- The file is missing many observations.
- The total deaths variable is missing all non-New York City deaths.
- It solved the problem.
- You receive the warning "Missing values created from missing values".

5.2 Merging and Performing Operations on Datasets

```
data MERGE1; set Ch5demo1  Ch5demo2; run;
```

- 185 observations were read in from Ch5demo1.
- 1,421 observations were read from Ch5demo2.
- There are 1,606 observations in the new data set.

- It appears correct.
- You could look at the original data set using PROC CONTENTS.

```
proc sort data=MERGE1; by zip; proc sort data= Ch5demo3; by zip;
run;

data MERGE2; merge MERGE1 ch5demo3; by zip; run;

data MERGE2_calcs; set MERGE2; tot_death = male_911_deaths +
female_911_deaths; death_rate = (tot_death / pop_tot)* 100000;
run;

data MERGE2_calcs; set MERGE2_calcs; work_pop = pop_2024
+ pop_2534 + pop_3544 + pop_4554; work_rate = (tot_death /
work_pop)* 100000; run;
```

- You need to sort the data sets before merging.
- The death rate for 11,566 is $16.8/100,000$; the deathrate for zip code for 10032 is $18.8/100,000$.
- ZIP Code 10032 is at increased risk perhaps because it is closer to ground zero.
- The new rates for workers are 34.6 for ZIP code 11566, and 35.1 for 10032.
- The rates went up because the denominator of working individuals is smaller. The rates based on the working population are probably closer to the "true" risk.

Chapter 6

(You may need to refer back to other chapters to complete some of these problems)

6.1 Reading in the Data Set

```
DATA NYCSPARCS; /* name the data set in work directory */
INFILE \nycsparcs.TXT' MISSOVER LRECl=452; * OBS=15000;
/* notice how we commented out the obs=, will read all the
   observations */INPUT /* input the variables you are
      interested in */
@18   DATE yymmn6.
@44   AGE       3.
@50   COUNTY    $CHAR2.
@52   ZIP       $CHAR5.
@57   SEX       $CHAR1.
@58   RACE      $CHAR2.
@71   PDX       $CHAR6.
@349  LOS       4.
;
```

6.2 Creating New Variables

```
/**************CREATE MONTH VARIABLE***********/
month=MONTH(DATE);
/***********CREATE NUMERIC GENDER VARIABLE***************/
IF sex='M' then male=1;
   else male=0;
IF sex='F' then female=1;
           else female=0;

/**********CREATE NUMERIC VARIABLES FOR RACE ***********/
IF RACE = '01' then White=1;
               else White=0;
IF RACE = '02' then Black=1;
               else Black=0;
IF RACE = '04' then Asian=1;
               else Asian=0;
IF RACE = '88' then Other_Race=1;
               else Other_Race=0;
IF RACE = '99' then Unknown_Race=1;
               else Unknown_Race=0;

/************MORTALITY *************************/
IF DISPO = '20'  then death=1;
                 else death=0;

/*********** SUBSTANCE ABUSE ****************/

if pdx in /* use ICD9 codes to create diagnoses */
('2910','2911','2912','2913','2914','2915','29181','29189',
'2919','2920','29211','29212',
'2922','29281','29282','29283','29284','29289','2929',
'30300','30301','30302','30303','30390','30391','30392',
'30393','30400'
'30401','30402','30403','30410','30411','30412','30413','30420',
'30421','30422','30423','30430','30431','30432','30433','30440',
'30441','30442','30443','30450','30451','30452','30453','30460',
'30461','30462','30463','30470','30471','30472','30473','30480',
'30481','30482','30483','30490','30491','30492','30493','30500',
'30501','30502','30503','3051','30520','30521','30522','30523',
'30530','30531','30532','30533','30540','30541','30542','30543',
'30550','30551','30552','30553','30560','30561','30562','30563',
'30570','30571','30572','30573','30580','30581','30582','30583',
'30590','30591','30592','30593')
Then subst_ab=1;
Else subst_ab=0;

RUN;

PROC CONTENTS DATA=NYCSPARCS; /* check your file was read in */
RUN;
```

6.3 Using PROC MEANS

```
proc means data=nycsparcs;
var age;
run;

proc means data=nycsparcs;
var male;
run;
```

- There is a person aged 109 years old.

6.4 Using PROC FREQ

```
proc freq data=nycsparcs;
tables county race sex;
run;
```

- There is no real difference between numeric gender and character sex, although the numeric variable might be more useful if we need a dummy variable later on.
- Everyone came from a few counties, but since this is New York City data that makes sense. We refer to data documentation to determine that county 58 is the Bronx, 59 is Brooklyn, 60 is Manhattan, 61 is Queens, and 62 is Staten Island. The largest percentage of non-New York City residents comes from county 55 (Westchester).
- Over a quarter of the entries have race as other or unknown. This raises concern about the reliability and validity of that variable.

6.5 Using PROC TABULATE

```
proc tabulate data=nycsparcs; /*create output data set from
  tabulate procedure */
         where county in ('58' '59' '60' '61' '62');
         class month;
     var subst_ab;
     table month, subst_ab*sum subst_ab*pctsum;
ods output table=subst; /* note creating a table from output,
 will use it later to graph, can see it in explorer */
run;
```

- Brooklyn appears to need more resources.
- Staten Island may require less.
- These results may be affected by the number of hospital beds in a county, the number of beds available for substance abuse, and local medical practice in admitting or discharging substance abuse, the proportion of age groups in a county that might be more or less likely to abuse drugs.
- We would want to determine population-based age-stratified rates.

Chapter 7

7.1 PROC GCHART

```
proc gchart data=nycsparcs;
     vbar race;
     hbar race;
run;

proc gchart data=nycsparcs;
     vbar race / sumvar=los type=mean;
     hbar race / sumvar=los type=mean;
run;
quit;

proc gchart data=nycsparcs;
     pie race / sumvar=los type=mean
                fill=x explode='02';
run;
quit;
```

One advantage of hbar is that it returns statistics.

7.2 PROC GPLOT

```
proc tabulate data=nycsparcs; /*create output data set from
 tabulate procedure */
         where county in ('58' '59' '60' '61' '62');
         class month;
     var subst_ab;
     table month, subst_ab*sum subst_ab*pctsum;
ods output table=subst; /* note creating a table from output,
        we'll use it later to graph, can see it in explorer */
run;


proc gplot data=subst; /* note using output table from tabulate*/
     plot subst_ab_Sum * month ;
 symbol value=diamond i=spline
      c=red w=5;
run;
quit;
```

- The chart is not zeroed on the vertical axis, which makes changes look more impressive.
- The spline makes it look like a continuous process when it very much may not be so.
- You can use the "vaxis" option after the plot statement to define axis. It might be better to just join the points instead of spline interpolation. (The change in colors is just for fun.)

```
proc gplot data=subst;
     plot subst_ab_Sum * month / vaxis= 0 to 5000 by 200;
 symbol value=diamond i=join
      c=blue w=2;
run;
quit;
```

Chapter 8

8.1 One-Way Frequencies

```
proc print data=car;
   var type region safety weight;
run;

proc freq data=car;
   tables region safety type;
run;
```

Partial PROC PRINT Output

Obs	type	region	safety	weight
1	Medium	N America	0	3.395
2	Sport/Utility	N America	0	4.180
3	Medium	N America	0	3.145
4	Small	N America	0	2.600
5	Medium	N America	0	3.085
6	Medium	N America	0	2.910
7	Sport/Utility	N America	0	4.180
8	Medium	Asia	0	3.415
9	Medium	N America	0	3.995
10	Small	N America	0	2.600
11	Small	N America	1	2.765
12	Small	Asia	0	2.665
13	Medium	N America	0	3.100
14	Medium	N America	0	3.455
15	Medium	N America	0	3.055
16	Large	N America	0	3.450
17	Large	N America	0	3.640
18	Large	N America	0	4.195
19	Large	N America	0	3.985
20	Large	N America	0	4.480

- Measurement scale of each variable:
  - Safety—OrdinaL
  - Type—NOMINAL
  - Region—nominal
  - Weight—CONTINUOUS

- The proportion of cars built in North America is 0.6354.
- There are no unusual data values that warrant further attention.

8.2 Cross Tabulations

```
proc format;
   value safdesc 0='Average or Above'
                 1='Below Average';
run;

proc freq data=car;
   tables region*safety / expected cellchi2;
   format safety safdesc.;
run;
```

- For the cars made in Asia, 42.86% had a below-average safety score.
- For the cars with an average or above safety score, 69.70% were made in North America.
- There seems to be an association between region and safety. A higher percentage (75.41 vs. 57.14) of cars from North America had a higher safety rating.
- The cell where region is Asia and safety is below average contributed the most to any possible association.

8.3 Chi-Square

```
proc freq data=car;
   tables region*safety / chisq;
   format safety safdesc.;
run;
```

You fail to reject the null hypothesis that there is not an association. The p-value represents the probability of observing a chi-square value at least as large as the one actually observed, given that the null hypothesis is true.

8.4 Spearman

```
data car2;
   set car;
   size=1*(type='Sports' or type='Small') +
        2*(type='Medium') +
        3*(type='Large' or type='Sport/Utility');
run;


proc format;
   value sizename 1=Small
                  2=Medium
                  3=Large;
run;
```

```
proc freq data=car2;
   tables size*safety / chisq measures cl;
   format safety safdesc.
          size sizename.;
run;
```

- You should use the Mantel-Haenszel test to detect an ordinal association between size and safety.
- You reject the null hypothesis that there is not an ordinal association.
- The Spearman correlation statistic indicates that an ordinal association of moderate strength exists (0.5425) between size and safety.
- The 95% confidence interval around that statistic is (0.6932, 0.3917).

Chapter 9

9.1 Contingency Table Analysis

- The row mean scores difference statistic can be used to measure the evidence of an association between type by safety?
- With a p-value less than 0.0001, there is statistical evidence of an association between type by safety.
- You could use the uncertainty coefficient to measure the strength of the association between type by safety.
- The proportion of variability in the response variable that is explained by the predictor variable is 0.3011.

9.2 Stratified Analysis

```
   tables type*safety region*safety / all;
run;
```

- Use the MH statistic to detect an association between type by safety controlling for region.
- There is a statistically significant association of type with safety, holding region constant.
- Because there are no observations for "large" vehicles.
- Yes. There may be cells with zero in the denominator leading to undefined results.

```
proc freq data=sasuser.safety;
   tables type*region*safety / all bdt;
   exact or comor;
run;
```

- Yes. The adjusted odds ratio for the association of region and safety controlling for type differs from the crude or unadjusted odds ratio.
- The Breslow–Day statistic is not statistically significant, indicating there is no interaction between type and region.

Chapter 10

10.1 PROC MEANS

- SAS used all observations, most of which were coded zero because not recorded for adults.
- $N = 200,000$, $\mu = 318.3755000$.
- 134,412 children less than 1 year old were discharged from NYC hospitals in this year.
- $N = 22,611$, $\mu = 2816.11$.

```
DATA infants; /* name the data set in work directory */
INFILE 'C:\Users\Charlie\Documents\Columbia\Epi\SAS COURSE\Data
 Sets\nycsparcs.TXT' MISSOVER LRECl=452; * OBS=15000; /*
  notice how we commented out the obs=,
will read all the observations */
INPUT /* input the variables you are interested in */
@18   DATE yymmn6.
@44   AGE        3.
@50   COUNTY     $CHAR2.
@52   ZIP        $CHAR5.
@57   SEX        $CHAR1.
@58   RACE       $CHAR2.
@60   ETHNIC     1.
@61   SOURCE     $CHAR1. /*source of admission used with type of
                         admission to id prematures*/
@62   TYPE       $CHAR1. /* type of admission to identify
@71   PDX        $CHAR6.   newborns=type 04 */
@294  BIRTHWT    4.
@343  DISPO      $CHAR2.
@349  LOS        4.
@434  CHARGE     12.;

/*******************MORTALITY *************************/
IF DISPO = '20'  then death=1;
                 else death=0;
/********************** INFANT MORTALITY ***************/
IF AGE LT 1 and DISPO='20'  then infant_mort=1;
                            else infant_mort=0;
/********************** LOW BIRTH WEIGHT *************/
IF TYPE='4' and BIRTHWT LT 2500 then LBTWT=1;
                                else LBTWT=0;
/********************** PREMATURE BIRTH **************/
IF TYPE='4' and SOURCE='2'  then  preemie=1;
                            else prememie=0;
run;

proc means data=infants;
var birthwt;
run;
```

```
proc means data=infants n mean median std var clm;
var birthwt;
run;

data infants;
  set infants;
  if age lt 1;
  run;


proc means data= infants n mean median std var clm;
var birthwt;
run;
```

10.2 PROC UNIVARIATE

- The histogram and probability plot lead us to suspect the data are not normally distributed.
- The skewness statistic is −1.5, indicating the data are left skewed.
- The mean (2816) is less than the median (3200), again indicating left skewed.
- The kurtosis statistic is 1.3, indicating a high-peaked and heavy-tailed distribution.
- There are a lot of zeros, indicating either missing data or coding errors.
- Rerunning the analysis with zeros removed returns a more normal-appearing data distribution, with a mean closer to the median, less skewness, and a kurtosis value about the same as the previous run:

```
goptions reset=all fontres=presentation ftext=swissb htext=1.5;

proc univariate data infants mu0=2500 plot;
var birthwt;
histogram birthwt;
probplot birthwt / normal (mu=est sigma=est color=blue w=1) ;
run;


goptions reset=all fontres=presentation ftext=swissb htext=1.5;

proc univariate data infants mu0=2500 plot;
where birthwt ne 0;
var birthwt;
histogram birthwt;
probplot birthwt / normal (mu=est sigma=est color=blue w=1) ;
run;
```

10.3 PROC BOXPLOT

```
data infants;
set infants;
if county = '58' then borough = 1;
```

```
else if county ='59' then borough = 2;
else if county ='60' then borough = 3;
else if county = '61' then borough = 4;
else if county = '62' then borough = 5;
else borough = .;
run;

proc sort data=infants;
by borough;
run;

symbol color = salmon;
   title 'Boxplot NYC County Birthweights';
   proc boxplot data=infants04;
   where birthwt ne 0;
   plot birthwt*borough    / cframe   = vligb
                             cboxes   = dagr
                             cboxfill = ywh;
   run;
```

Chapter 11

11.1 PROC GLM

```
options ls=75 ps=45;  /* page and line spacing options*/
proc glm data = infants;
where birthwt ne 0;
       class borough;
       model birthwt=borough;
       means borough / hovtest;
       output out=check r=resid   p=pred;
run;
quit;  /* have to quit our of GLM */
```

The p-value for Levene's test is $p < 0.0001$; reject the null hypothesis of homogeneity

11.2 PROC GPLOT

```
/* now run gplot on the 'check' dataset created above*/
Proc gplot data=check;
         Plot resid*pred / haxis=axis1 vaxis=axis2 vref=0;
         axis1 w=2 major=(w=2) minor=none offset=(10pct);
         axis2 w=2 major=(w=2) minor=none;
         title 'plot residuals vs predictors';
run;
quit;
```

The residuals are evenly distributed about 0 with some evidence of outliers.

11.3 PROC UNIVARIATE

```
proc univariate data = check normal;
      var resid;
      histogram resid / normal;
      probplot;* resid / mu=est sigma=est color=blue w=1;
      title;
run;
```

- The mean is zero.
- It is consistent with the underlying assumptions for ANOVA.
- The kurtosis statistic is 3.3.
- The data are high peaked and heavy tailed.

11.4 LSMEANS

```
/* compare means */
proc glm data=infants;
   class borough;
   model birthwt=borough;
   lsmeans borough / pdiff=all adjust=tukey;
   title 'Data: Multiple Comparisons';
run;
quit;
```

- The lowest mean birth weight is 2,662 grams in borough 1. It is statistically significantly different from the other boroughs.
- Box plots would be helpful to compare the mean birth weights across boroughs.

Chapter 12

12.1 Read in the Pedestrian Injury Data Set

```
proc contents data=ch9.ch9exercise;
run;
```

12.2 Print Out the Data

```
proc print data=ch9.ch9exercise;
var numinj totpop perblack perhisp medhsinc pci;
id name;
run;
```

12.3 Create an Injury Rate Variable

```
data ch9exercise;
set ch9.ch9exercise;
injrate=(numinj/totpop)*1000;
run;
```

12.4 PROC UNIVARIATE

```
options ps=50 ls=76;
goptions reset=all fontres=presentation ftext=swissb htext=1.5;
proc univariate data=ch9exercise;
   var injrate perblack perhisp medhsinc pci;
   histogram injrate perblack perhisp medhsinc pci/ normal;
   probplot  injrate perblack perhisp medhsinc pci
             / normal (mu=est sigma=est color=red w=2);
   id name;
   title 'Univariate Statistics of pedestrian injury data set';
run;
```

- The average number of injuries per 1,000 population for a Nassau County ZCTA community during the observation period was 2.
- Hempstead had the highest injury rate, six injuries per 1,000 population.
- Freeport and Hempstead had the highest proportion of Hispanic residents.
- Inwood and Hempstead had the the lowest median household income.
- "perblack" and "perhisp" appear to be least likely to be normally distributed.

12.5 Log Transformation

```
data ch9exercise;
set ch9exercise;
lnblack=log(perblack);
lnhisp=log(perhisp);
run;

proc univariate data=ch9exercise;
   var lnblack lnhisp;
   histogram lnblack lnhisp/ normal;
   probplot  lnblack lnhisp
             / normal (mu=est sigma=est color=red w=2);
   id name;
   title '';
run;
```

Yes.

12.6 PROC GPLOT

```
/*Use the following syntax to set up the options for
 your plots and define axes*/

Options ps=50 ls=64;
Goptions reset=all gunit=pct border
Fontres=presentation ftext=swissb;

Axis1 length=70 w=3 color=blue label=(h=3) value=(h=3);
Axis 2 length=70 w=3 color=blue label=(h=3) value=(h=3);

/*Invoke the above options and axes by
including the following line after your plot statement*/

/ vaxis=axis1 haxis=axis2

Options ps=50 ls=64;
Goptions reset=all gunit=pct border
               Fontres=presentation ftext=swissb;

Axis1 length=70 w=3 color=blue label=(h=3) value=(h=3);
Axis 2 length=70 w=3 color=blue label=(h=3) value=(h=3);

proc gplot data=ch9exercise;
   plot injrate * (lnblack lnhisp medhsinc pci)
        / vaxis=axis1 haxis=axis2;
   symbol1 v=dot h=2 w=4 color=red;
   title h=3 color=green
         'Scatter Plot of Pedestrian Injury Rate by Explanatory
           Variables';
run;
quit;
```

- "perhisp" seems to most demonstrate a relationship with injury rate.
- The plot would best be described as a direct linear relationship.

12.7 PROC CORR

```
proc corr data=ch9exercise rank;
   var lnblack lnhisp medhsinc pci;
   with injrate;
   title '';
run;
```

- "perhisp" was most strongly correlated with pedestrian injury rate.
- The relationship between income in a community and pedestrian injury rate was weakly negative.

- There is something in communities with large proportions of poor Hispanics or Blacks that put people at risk for pedestrian injuries.

Chapter 13

13.1 Infant Birth Weight and Hospital Charges: Assumptions

```
options ps=50 ls=76;
goptions reset=all fontres=presentation ftext=swissb htext=1.5;

proc univariate data=  infants normal;
var charge birthwt;
histogram charge birthwt;
probplot charge birthwt;
run;
```

The data do not appear normally distributed.

13.2 Infant Birth Weight and Hospital Charges: Correlation

```
proc corr data=infants rank;
   var charge;
   with birthwt;
   title 'PROC CORR: infant birthweight and charge';
run;
```

- There is a statistically significant relationship between birthweight and charges $r = -0.22, p < 0.0001$.
- The relationship is inversely proportional.
- We should look at it with a scatterplot.

```
Options ps=50 ls=64;
Goptions reset=all gunit=pct border
               Fontres=presentation ftext=swissb;

Axis1 length=70 w=3 color=blue label=(h=3) value=(h=3);
Axis 2 length=70 w=3 color=blue label=(h=3) value=(h=3);

proc gplot data= infants;
   plot charge * birthwt
        / vaxis=axis1 haxis=axis2;
   symbol1 v=dot h=2 w=4 color=red;
   title h=3 color=green
         'Plot of infant birthweight and charge';
run;
```

13.3 Infant Birth Weight and Hospital Charges: Linear Regression

```
proc reg data=infants;
   where birthwt ne 0;
   model charge=birthwt;
   title 'Simple Linear Regression of charges and birthweight';
run;
quit;
```

- The overall F test is 3497 $p < 0.0001$.
- Birthweight is a statistically significant predictor of total charges $p < 0.0001$.
- The model explains 14% of the variability $R^2 = 0.15$.
- Total explained variance is the square of Pearson's, correlation coefficient.

```
quit;
```

Chapter 14

14.2 Residuals

```
options ps=50 ls=97;
goptions reset=all fontres=presentation ftext=swissb htext=1.5;
proc reg data=sasuser.b_grades;
   model gpa=score;
   plot r.*(p. score);
   plot student.*obs. / vref=3 2 -2 -3
                        haxis=1 to 25 by 1;

   title 'Residual Diagnostic Plots';
run;
quit;
```

Index

C. DiMaggio, *SAS for Epidemiologists: Applications and Methods*,
DOI 10.1007/978-1-4614-4854-9,

R