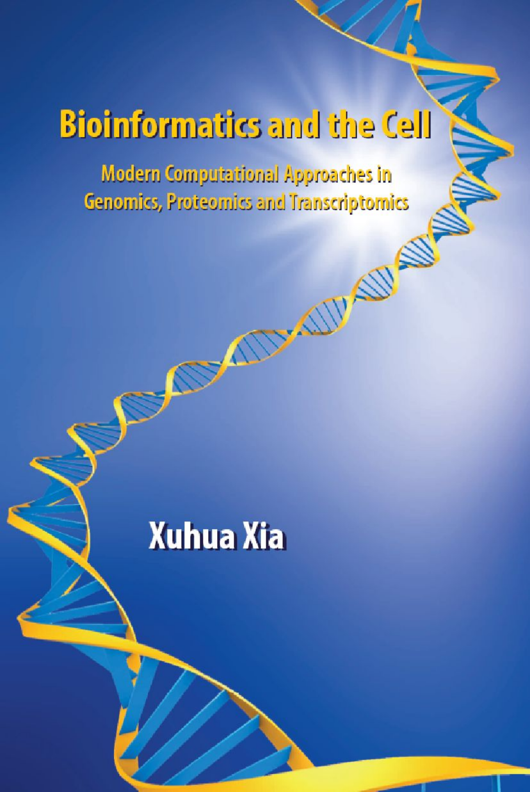


Bioinformatics and the Cell

Modern Computational Approaches in Genomics, Proteomics and Transcriptomics

Xuhua Xia

BIOINFORMATICS AND THE CELL
Modern Computational Approaches in Genomics, Proteomics and Transcriptomics

BIOINFORMATICS AND THE CELL
Modern Computational Approaches in Genomics, Proteomics and Transcriptomics

by
Xuhua Xia
University of Ottawa, Ontario, Canada

Springer

Xuhua Xia
CAREG and Biology Department
University of Ottawa
30 Marie Curie, P.O. Box 450
Ottawa, Ontario K1N 6N5
Canada

Library of Congress Control Number: 2007922341

ISBN-10: 0-387-71336-0 e-ISBN-10: 0-387-71337-9
ISBN-13: 978-0-387-71336-6 e-ISBN-13: 978-0-387-71337-3

Printed on acid-free paper.

9 8 7 6 5 4 3 2 1

springer.com

CONTENTS

PREFACE

Biological and biomedical sciences are becoming more interdisciplinary, and scientists of the future need interdisciplinary training instead of the conventional disciplinary training. Just as Sean Eddy (2005) wisely pointed out that sending monolingual diplomats to the United Nations may not enhance international collaborations, combining strictly disciplinary scientists trained in either mathematics, computational science or molecular biology will not create a productive interdisciplinary team ready to solve interdisciplinary problems.

Molecular biology is an interdisciplinary science back in its heyday, and founders of molecular biology were often interdisciplinary scientists. Indeed, Francis Crick considered himself as “a mixture of crystallographer, biophysicist, biochemist, and geneticist” (Crick, 1965). Because it was too cumbersome to explain to people that he was such a mixture, the term “molecular biologist” came handy. To get the crystallographer, biophysicist, biochemist, and geneticist within himself to collaborate with each other probably worked better than a team with a crystallographer, a biophysicist, a biochemist and a geneticist who may not even be interested in each other’s problems.

Bioinformatics was born in response to the interdisciplinary demand of modern biological and biomedical sciences as a joint effort by among mathematicians, computational scientists and biologists of all colors and shades. It is a peculiar branch of science. A conventional branch of science such as quantum mechanics in physics or population genetics in biology will typically have a few classic publications laying down its theoretical foundation, delineating its boundary and interface with other related sciences, formulating its central questions, and highlighting the spectacular views within and beyond that particular mansion of science. Once the mansion has been skeletally constructed, subsequent works will only serve to beautify the mansion but will not alter the general structure of the mansion which remains easily recognizable by people within the mansion and those in its neighbourhood. There is little controversy as to how the mansion looks, even when viewed from different perspectives.

Bioinformatics is different. I have been told that bioinformatics was not built by one or a few visionary giants of science, and that it is not even a single mansion. Instead, it is the Wild West before the law arrives (Eddy, 2005), dotted by a large number of trailer houses or even temporary tents that have been built and found workable elsewhere in the kingdom of science. Many people living in this rough terrain do not know where they belong but, after living here for some time, found it necessary to give the dwelling a label. When someone murmured the word "bioinformatics", everyone thought it a godsend and the town of bioinformatics was born, and the inhabitants begin to call themselves bioinformaticians.

Bioinformaticians differ dramatically in their views and their descriptions of their town. This is partially reflected in the flagship journal of the field, Bioinformatics. Most papers in the journal were treated by conventional biologists as Wild West stories and ignored with a passion, except for a few that get a great deal of attention and citation by proclaiming the finding of gold. The only consensus among bioinformaticians seems to be that bioinformatics deals with very big computational problems. However, when asked about what the very big problems are, most bioinformaticians, to paraphrase Peter Medawar, will become instantly solemn and shifty-eyed, solemn because they think that they have something profound to declare, and shifty-eyed because they really have nothing to declare.

Such a perception of bioinformatics is witty but unfair, because bioinformatics does have a root to trace to and a central theme with a focus. For many years, a challenging question near and dear to the mind of many leading biologists is how living cells work. A living cell is a system with cellular components interacting with each other and with extracellular environment, and these interactions determine the fate of the cell, e.g., whether a stem cell is going to become a liver cell, a brain cell, or a cancerous cell. It then became quite obvious that, to understand how living cells work, those cellular components and their interactions would need to be identified and characterized. The most important cellular components happened to be universally acknowledged to be the genome, the transcripts and the proteins. The characterization and analysis of these three types of cellular components leads to genomics, transcriptomics and proteomics that jointly drive the development of bioinformatics.

Genomics leads to two developments. The first is to allow a much faster identification of proteins by combining mass spectrometry data with genomic databases. Second, genomic sequences have enabled SAGE (sequential analysis of gene expression) and microarray technology which have spawned transcriptomics which is a synonym of functional genomics. Biologists can now routinely monitor the gene expression at the genomic scale over time or compare gene expression between control cells and treatment cells or along the developmental path of a particular cell type.

There are two major problems with the transcriptomic data. The first is that the relative abundance of transcripts as characterized by SAGE or microarray experiments is not always a good predictor of the relative abundance of proteins, yet proteins are true workhorses in the cell. Many proteins that are produced as a result of alternative splicing and posttranslational modification will not reveal their mystery in our analysis of transcriptomic data. It should be quite clear that, in order to characterize the cellular components and their interactions, one needs the corroboration of proteomic, genomic and transcriptomic data.

At this point, one is tempted to conclude that bioinformatics has three facets labelled proteomics, genomics and transcriptomics and that it deals mainly with characterizing cellular components and their interactions. But bioinformatics goes beyond this. Genes and genomes have evolved from time immemorial, as do interactions among genes and gene products. The genomic change is particularly well exemplified by the infectious diseases caused by the influenza viruses, the SARS virus, and HIV, all evolving quickly as a result of mutation, recombination and selection. Studying the dynamic nature of genes and genomes, tracing their phylogenetic relationships and reconstructing their ancestral states allow biologists to gain the advantage perceived by Aristotle thousands of years ago, i.e., “He who sees things grow from the very beginning has the most advantageous view of them.” For this reason, molecular evolution is now an essential component of bioinformatics.

Many books have now been written on bioinformatics. They tend to fall on two extremes. In one extreme are books featuring computational details with a great deal of mathematics (e.g., Pevzner, 2000), while in the other extreme one finds books treating bioinformatics mostly as a giant black box (e.g., Baxevanis and Ouellette, 2005). The former is for computational scientists and mathematicians who, after reading the book, will remain computational scientists and mathematicians. The latter is for biologists who, after reading the book, will remain biologists. Such books often have limited contribution to creating interdisciplinary scientists needed in modern biological and biomedical sciences.

Most biologists cannot appreciate the beauty of mathematics without having the equations rendered to numbers. Remarkably, neither can mathematicians and computational scientists appreciate the beauty of biology without having the cells and bugs rendered to numbers. This book is my effort to render both mathematical equations and biology to numbers. It is aimed at creating truly interdisciplinary scientists to prosper in the Wild West of bioinformatics.

Although the book covers bioinformatics methods at a level more advanced than most other bioinformatics books, the extensive numerical illustration of these methods should make it accessible to most senior undergraduate students and graduate students majoring in science and software engineering. An additional advantage of using it as a textbook is that nearly all algorithms in the book are implemented in a free and user-friendly computer program (DAMBE).

Practising biologists with reasonably good programming skills should be able to implement most algorithms themselves and check the output against numerically illustrated examples in the book. They should soon find such learning experience intellectually rewarding and mentally satisfying. Some of them might even be pleasantly surprised to learn that they can quickly create a much needed computational method that their computer technicians have failed to create in a whole year.

I have tried my best to “make everything as simple as possible, but not simpler”. While most numerical illustrations of advanced computational algorithms in this book are toy examples, they require only simple extensions to tackle real data. To paraphrase the late C. C. Li, it is not necessary to create a rainbow spanning the sky to demonstrate how a rainbow forms – a small one is convincing enough.

“Please read the book”.

ACKNOWLEDGEMENT

It is said that only a military general fielding an army in the frontier could truly appreciate and fully acknowledge the contribution of the comrades in arms protecting his left and right, and the logistics department backing him up from behind. Not being a general and having never been close to becoming one, it seems inevitable that my acknowledgement is to be inadequate and defective.

But I do have comrades in arms protecting me and friends and family members helping me with logistics. I am fortunate to have Stephan Aris-Brosou, Linda Bonen, Andre Dabrowski, Christian Detellier, Marc Ekker, Guy Drouin, Donal Hickey, Stan Matwin, Youlian Pan, Steve Perry, Marcel Turcotte, Vance Trudeau, Morris Zhang, and David Zhou as colleagues and friends who have helped me not only in the role of comrades in arms but also as an essential component of my logistics department.

Much of the book is based on my two undergraduate courses and a graduate course on bioinformatics and molecular evolution at University of Ottawa. My students, especially graduate students, have supplied me with constant challenges and pedagogical insights. The following graduate students have commented on various chapters and corrected typing errors: Malisa Carullo, Sam Khalouei, Pinchao, Ma, Jan Mennigen, Gareth Polidwor, Jason Popescu, Ziyu Song, Huiling Xiong, Xiaoquan Yao.

The internet has helped me keep contact with colleagues far and wide. Many concepts and methods in the book were included as a consequence of my discussions with Esther Betran, Adam Eyre-Walker, Joe Felsenstein, Youngbi Fu, Olivier Gascuel, Brian Golding, Paul Higgs, King Jordan, Sudhir Kumar, Wen-Hsiung Li, Jean Lobry, Manyuan Long, Axel Meyer, Etsuko Moriyama, Eduardo Rocha, Eduardo Roman, Marco Salemi, Anne-Mieke Vandamme, and Reiner Veitia.

While many people have read and commented on some parts of the book, Stephan Aris-Brosou and Yongbi Fu have read almost every chapter and provided detailed comments and suggestions. Joe Felsenstein suggested, upon a quick reading of an earlier draft, adding the EM algorithm for maximum likelihood calculation and MCMC algorithm for evaluating posterior probabilities in Bayesian inference. As a little known fact, it is also Joe who, together with Ben Hall, got me started with molecular evolution and phylogenetics when I developed severe allergic response to deer mice that I used to study.

The patience of my editors, Joseph Burns and Marcia Kidston, are much appreciated. As an embarrassing manifestation of my naivety, I promised, in late 2003, to deliver the book to the publisher before February 15, 2005, and I thought it was a conservative estimate. After all, I had been teaching these computational methods for many years and producing a book seemed to require little more than simply dumping my lectures to a printer. What a humbling experience in writing the book!

I also wish to take this opportunity to thank my wife, Zheng, my daughter Kimberley, and my sons Jeffery, Jeremy and Jadon, for their love, support and entertainment. Without their constant demand for walks, talks and outdoor activities, Daddy would probably have spent all his time in front of computers and consequently ruined his health many years ago.

A family of increasing size has helped me better appreciate the importance of financial matters, so that I will not forget to acknowledge the grant support I have received from University of Ottawa and from Canadian National Science and Engineering Research Council's Discovery Grant, Research Tools and Instrument Grant, and Strategic Grant for doing research in bioinformatics and molecular evolution. While our funding agencies will inevitably focus on research projects aiming to turn the white and light from the sky into the golden and heavy in the bank, I am glad that there is still some leftover for other enquiries.

Chapter 1

BLAST AND FASTA

Mathematics in string searching algorithms

1. INTRODUCTION

One might find it odd to start a book on bioinformatics and the cell with BLAST and FASTA and I have received two kinds of objections from readers of the book draft. The first argued that any book on a new subject should provide some historical background so that the reader can position himself or herself in proper historical context. The following one-paragraph history of genomics, transcriptomics and proteomics is added in response to this objection.

Genomics is often considered to have started in 1986 when two significant events happened. First, the U. S. Department of Energy (DOE) announced the Human Genome Initiative aiming to produce a reference human genome sequence. Second, Leroy Hood developed the first automatic DNA sequencer, which paved the way for the official beginning of the Human Genome Project in 1990. The year 1995 witnessed the completion of the first three bacterial genome sequencing projects, with the *Haemophilus influenzae* Rd KW20 genome in July (Fleischmann *et al.*, 1995), the *Synechocystis sp.* PCC 6803 genome in August (Kaneko *et al.*, 1996; Kaneko *et al.*, 1995), and the *Mycoplasma genitalium* G-37 genome in October (Fraser *et al.*, 1995). The first working draft of the entire human genome was completed in 2000, soon to be followed by the simultaneous publication of the human genome in *Nature* and *Science* in February, 2001 (Lander *et al.*, 2001b; Venter *et al.*, 2001). Transcriptomics started mainly with the development of gene arrays such as macroarrays (Chuang *et al.*, 1993; Tao *et al.*, 1999) and microarrays (Schena, 1996; Schena, 2003) and serial analysis of gene expression (Madden *et al.*, 1997; Saha *et al.*, 2002; Velculescu *et al.*, 1995; Velculescu *et al.*, 1997; Zhang *et al.*, 1997). The terms "proteome" and "proteomics" were coined by Marc Wilkins and colleagues in 1994 (Ezzell, 2002), and large-scale proteomic research started with sodium dodecyl sulfate polyacrylamide gel electrophoresis, SDS-PAGE (Laemmli, 1970). Subsequent perfection of isoelectric focusing leads to the most frequently used protein separation method, the 2D-SDS-PAGE. Large-scale peptide analysis methods have been developed by John Yates and colleagues (Washburn *et al.*, 2001; Yates, 2004a, 2004b). Mass spectrometry used in combination with affinity purification and/or chemical cross-linking has made significant contributions to protein interaction networks (Figeys, 2003b, 2003a; Vasilescu and Figeys, 2006). Protein arrays have recently been developed to directly assess protein-protein interactions (Figeys, 2002; Sloane *et al.*, 2002; Wilson and Nock, 2002). The characterization and analysis of genomes, transcriptomes and proteomes have driven the development of bioinformatics and resulted in many new insights in understanding how living cells function (or fail to function as in cancer).

The second objection against starting the book with BLAST and FASTA is based on (1) the approach might mislead the reader to think that this book is all about something everybody knows and (2) few people would agree, with good reasons, that any branch of science starts with a homology search algorithm. If one is to write a book progressing from genomics to transcriptomics and proteomics, then naturally one should start with sequencing genomic fragments, contig assembly to assemble the sequence fragments to a contiguous genome, and sequence annotation to show the location of biologically interesting genes and motifs on the genome. This should then be followed by the characterization and analysis of transcripts and proteins in the cells, and finally bring the reader to a new horizon with a new concept and perception of a living cell. Why start with BLAST and FASTA?

The reasoning above is indeed eloquent and compelling, but it is unfortunately against an important pedagogical principle, i.e., any theme of presentation should begin with a common, ideally universal, entry point. This pedagogical principle points unambiguously to BLAST and FASTA. Many colleagues of mine share the opinion that BLAST and FASTA in bioinformatics are equivalent to PCR in molecular biology wet labs and deserve more recognition.

Yet my choice of starting with BLAST and FASTA is not just because of the pedagogical principle, but more because of two other reasons. First, the mathematics and computation underlying these two widely used computational tools represent the common denominator of mathematics and algorithms in many other bioinformatics tools. Second, while a large number of researchers use BLAST and FASTA daily, very few actually understand the statistical and computational aspects of them. A simple but systematic presentation of the mathematical, statistical and computational aspects of these bioinformatics tools will not only help researchers to better interpret their BLAST and FASTA output and implement these methods in their own programs, but also foster better appreciation of bioinformatics as an interdisciplinary science and of its major players - in this case David Lipman, Samuel Karlin, Stephen Altschul, William Pearson and many of their colleagues.

BLAST and FASTA belong to the category of sequence search and annotation tools. Once genomic scientists obtain a long contig or a genomic sequence by contig assembly, the next step is obviously to find out what the long stretch of A, C, G, T means. This is the subject of sequence annotation. Sequence annotation is perhaps the most misunderstood subject. Laypersons may equate sequence annotation to adding personal scribbles to the margin of a novel, without realizing that a great deal of mathematics and computation, as well as a great deal of biology, is needed to do the job.

The pivotal component of sequence annotation is gene finding. There are two major categories of computational tools for gene-finding. The first is based on known genes in molecular databases, and uses homology search tools such as FASTA (Pearson and Lipman, 1988) and BLAST (Altschul *et al.*, 1990; Altschul *et al.*, 1997). The second, better known as gene prediction, is based on known gene structures, and represented by GENSCAN (Burge and Karlin, 1997). Existing software for gene-finding often combine both approaches, e.g., GenMark (Hayes and Borodovsky, 1998), GLIMMER (Salzberg *et al.*, 1998), Orpheus (Frishman *et al.*, 1998), Projector (Meyer and Durbin, 2004) and YACOP (Tech and Merkl, 2003).

The methods for finding sequence similarities caught the attention of biologists when an oncogene (i.e., a gene in a virus that causes a cancer-like transformation of the infected cell), v-sys, was found to be similar to PDGF, the platelet-derived growth factor (Doolittle *et al.*, 1983; Waterfield *et al.*, 1983). The increasing number of genes and genomes deposited in GenBank (Benson *et al.*, 2005) implies increasing importance of methods for finding genes by homology search. Indeed, sequence similarity search has been claimed to be the most effective method for exploiting the information in the rapidly growing molecular sequence databases (Pearson, 1998).

Conventional methods for similarity searchers are based on local sequence alignment using dynamic programming (Smith and Waterman, 1981a). For a given scoring scheme, such methods will guarantee the finding of the optimal alignment. However, such methods are very slow. FASTA and BLAST use heuristic methods for similarity search. They may miss homologous sequences, but are very fast. With terabytes of molecular sequences in the database to search through, the speed becomes more important than sensitivity of homology detection.

A similarity search will generate a similarity score, which needs to be evaluated for its statistical significance. BLAST became more popular than FASTA partially because the early versions of FASTA did not evaluate the statistical significance of the resulting sequence matches. Latter versions of FASTA have incorporated such evaluations.

We will first offer a simple but detailed presentation of the mathematics involving string matching, which allows us to design a filter to eliminate a large number of insignificant matches. The string search algorithms used in FASTA and BLAST are then presented at a level to allow programmers to implement the algorithms in their own software.

2. MATHEMATICS OF STRING MATCHING

2.1 Basic concepts

Given P_A, P_C, P_G, P_T in a target (database) sequence, the probability of a query sequence (Q), say, ATTGCC, having a perfect match of the target sequence (D) is:

$$p = P_A P_C^2 P_G P_T^2 \qquad (1.1)$$

Let L_D be the target sequence length and L_Q be the query sequence length, the number of possible "matching operations", i.e., number of times one can shift Q against D in search for a perfect match of L_Q letters, is

$$n = (L_D - L_Q + 1) \qquad (1.2)$$

For example, with Q = ATG and D = CGATTGCCCG, $L_Q = 3$, $L_D = 10$, n = 8.

The probability distribution of the number of matches follows (approximately) a binomial distribution with p defined in Eq. (1.1), n defined in (1.2), and q = 1 - p:

$$(p+q)^n = p^n + \frac{n!}{(n-1)!1!}p^{n-1}q + \ldots + \frac{n!}{x!(n-x)!}p^x q^{n-x} + \ldots + q^n$$
$$p(x) = \frac{n!}{x!(n-x)!}p^x q^{n-x} \tag{1.3}$$

If $P_A = P_C = P_G = P_T = 0.25$, then $p = 0.25^3 = 0.015625$, and $q = 1 - p = 0.984375$. With $L_Q = 3$, $L_D = 10$, we have $n = 8$. So the probabilities of having 0, 1, 2, …, r exact matches of three letters are $p(0) = 0.881626$, $p(1) = 0.111953$, $p(2) = 0.00622$ and so on. The probability of having at least one match is simply $1 - p(0) = 0.118373565$.

Binomial distribution is troublesome in computation when n is large because computers will have overflow errors to get the factorial with a large n. When $np < 1$ and n is very large, the binomial distribution can be approximated by the Poisson distribution with mean and variance equal to np. The mathematical detail of converting the binomial probability distribution to the Poisson distribution is shown below:

$$\begin{aligned} p(x) &= \frac{n!}{(n-x)!x!}p^x q^{n-x} = \frac{n!}{(n-x)!x!}p^x q^{-x} q^n \\ &= \frac{n(n-1)(n-2)\ldots(n-x+1)}{x!}\left(\frac{p}{q}\right)^x (1-p)^n \\ &\approx \frac{n^x}{x!}p^x e^{-np} = \frac{n^x p^x}{x!}e^{-np} = \frac{(np)^x}{x!}e^{-np} = e^{-\lambda}\frac{\lambda^x}{x!} \end{aligned} \tag{1.4}$$

The approximation assumes a large n and a small p near 0, so that $(n - x + 1) \approx n$, $p/q \approx p/1 = p$, and $(1 - p)^n \approx e^{-np}$. The Poisson distribution has only one parameter λ, and P(x) is very easy to compute. To use the Poisson distribution to approximate the binomial example above with $n = 8$ and $p = 0.015625$, we have $\lambda = np = 0.125$, and $p(0) = 0.882496903$, $p(1) = 0.110312113$, $p(2) = 0.006894507$, and so on. The probability of having at least one match is simply $1 - p(0) = 0.117503097$. Although our n is not very large and p not very small, these values are still quite similar to those from the binomial distribution.

The simple mathematics concerning string matching has practical applications. For example, in serial analysis of gene expression or SAGE (Velculescu *et al.*, 1995), it is important to know how many transcribed RNA may not contain the recognition site (GTAC) of the NlaIII restriction enzyme and consequently would be missed by the method. Suppose the genome is slightly GC biased with $P_C = P_G = 0.3$ and $P_A = P_T = 0.2$. What is the probability that a transcribed RNA will not have the GTAC? In this case Q = GTAC and D is a specific transcribed RNA. Assuming L_D = 3000, we have n = (3000 – 4 +1), p = $0.3^2 \times 0.2^2$, and the Poisson parameter λ = 10.7892. The probability that a transcript does not have the restriction site is 0.000020621. However, if L_D = 300, the probability that a transcript does not have the restriction site is 0.343283. Thus, the SAGE method is strongly biased against short mRNA.

Another string matching problem in SAGE is whether a sequence fragment of 14 nucleotides are sufficient to identify a gene uniquely in a genome, say a human genome with 3×10^9 bp. With L_Q = 14 and L_D = 3×10^9 and assuming equal nucleotide frequencies of 0.25 in the human genome, we have the Poisson parameter λ = 11.17587085, and the probability of having at least one random match is 0.999986. Thus, a sequence fragment of 14 nucleotides cannot uniquely identify a gene product with typical eukaryotic genomes. This obvious fact is often not recognized. For example, the EMBL site (http://www.embl-heidelberg.de/info/sage/) on SAGE states that "But scientists don't need to read long sequences. They've discovered that just fourteen letters are enough to match a RNA to the precise gene that produced it." This is false. One can verify this easily by searching the tags in a published SAGE experiment characterizing human transcriptome (Velculescu *et al.*, 1999) against human protein-coding genes. Some tags have more than 100 matches, although to my knowledge no published paper based on SAGE experiment has mentioned such multiple-match tags. Fourteen letters are not enough, and long SAGE (Ryo *et al.*, 2000; Saha *et al.*, 2002) is necessary.

The examples above involve matching the entire query string Q against the target string D. Now we consider the problem of matching a substring of Q against a substring of D. We designate the length of the shared substring as L.

Given Q, D, L_Q and L_D, and assuming $P_A = P_C = P_G = P_T = 0.25$, the probability of finding an exact match of at least L ($L \le L_Q$ and $L \le L_D$) consecutive letters is $p = 0.25^L = 2^{-2L}$. The match of L consecutive letters between Q and D can happen at m = (L_Q – L +1) positions on Q and at n = (L_D – L +1) positions on D. You might have noted, in the BLAST output, terms such as "Effective length of query" and "Effective length of database". These terms are equivalent to m and n, respectively. There are mn possible matching operations, each with a probability of 0.25^L of getting a match of L consecutive letters. The expected number of matches with length at least L is therefore

$$E = mn0.25^L \tag{1.5}$$

Most BLAST web interfaces allow you to input a cutoff E-value. Suppose we are to BLAST sequence Q of length 10 against genomic sequence D of length 10,000,000. Does it make sense to set the E-value to 0.01? It does not, because we expect nearly 10 perfect matches of Q against D by random chance in this case. In other words, it is impossible for any returned match to have an E-value of 0.01. The smallest E-value among the returned matches (i.e., those perfect matches) will be nearly 10. A user-friendly implementation of the BLAST or FASTA algorithm would ignore the input E-value and return all perfect matches or at least display a tactful message saying that the user has committed a small and forgivable sin, but a strict implementation of the algorithm will simply return you with no match (A computational tool is user-friendly when the computer programmer works under the assumption that some users are fools).

In BLAST and FASTA literature, one may also encounter an equation in the following form

$$E = mne^{-\lambda R} \tag{1.6}$$

Eq. (1.6) and Eq. (1.5) are equivalent with λ = - ln(0.25) = 1.386294361 and R = L, where the value of 0.25 arises from the assumption of equal nucleotide frequencies.

Noting that $0.25 = 2^{-2}$, we can also rewrite Eq. (1.5) as

$$E = mn2^{-2L} = mn2^{-S} \tag{1.7}$$

where S = 2L. Eq. (1.7) has become particularly useful because it was found through computer simulations (Altschul, 1996; Altschul *et al.*, 1997; Pearson, 1998; Waterman and Vingron, 1994) that, when S is computed with a particular scoring scheme, Eq. (1.7) can be applied to situations involving two strings not only with consecutive matched letters, but also with mismatches and gaps.

Figure 1-1 shows the output from BLASTing a nucleotide query sequence against a local BLAST database containing all annotated *Mycoplasma genitalium* coding sequences. From the output (Figure 1-1) we see 35 matches, 3 mismatches, 1 gap and 2 gap extensions. The scoring scheme has the match score (M) equal to 1, mismatch score (MM) equal to -3, gap open penalty (G_o) equal to 5 and gap extension penalty (G_e) equal to 2 (Figure 1-1). When the raw score (R) is computed according to this scoring scheme, and the bit-score (S) is computed with R and two scaling factors λ and K, Eq. (1.7) can be used to obtain the E-value:

$$R = 35\times 1 + 3\times(-3) - 1\times 5 - 2\times 2 = 17$$

$$S = \frac{\lambda R - \ln(K)}{\ln(2)} = \frac{1.37\times 17 - \ln(0.711)}{\ln(2)} \approx 34.1 \qquad (1.8)$$

$$E = mn2^{-S} = 26\times 520557\times 2^{-S} = 0.000735$$

where λ and K, shown in every BLAST output, are scaling constants (Altschul *et al.*, 1997, and literature cited therein) estimated from computer simulation. Note that the E-value is calculated in the same way as in Eq. (1.7).

```
BLASTN 2.2.4 [Aug-26-2002]
...
Query= Seq1  38
Database: MgCDS
           480 sequences; 526,317 total letters
                                                    Score    E
Sequences producing significant alignments:        (bits) Value
MG001 1095  bases                                    34   7e-004
 Score = 34.2 bits (17), Expect = 7e-004
 Identities = 35/40 (87%), Gaps = 2/40 (5%)

Query: 1  atgaataacg--attatttccaacgacaaaacaaaaccac 38
           ||||||||||  |||||||||||  |||||| ||||||||
Sbjct: 1  atgaataacgttattatttccaataacaaaataaaaccac 40

Lambda         K        H
  1.37    0.711     1.31
Matrix: blastn matrix:1 -3
Gap Penalties: Existence: 5, Extension: 2
…
effective length of query: 26
effective length of database: 520,557
```

Figure 1-1. Output from a nucleotide BLAST query.

One may note that L in Eq. (1.5) is the same as R given M = 1, because R = L×M = L in case of exact string match with no mismatches or gaps involved. Because S = 2L in (1.7) and because L = R with ungapped match and M = 1, we have S = 2R. This relationship between S and R is similar to their relationship specified in the second equation in in Eq. (1.8). Note that R will increase with the length of the query and target sequences. With large R, S is approximately equal to 2R, i.e.,

$$S = \frac{\lambda R - \ln(K)}{\ln(2)} \approx 2R \text{ (with large R)} \qquad (1.9)$$

The E-value can be used as the λ parameter in the Poisson distribution in Eq. (1.4) to get the probability of having 0, 1, ..., x matches that are as good or better than the reported match. For our example, p(0) = 0.999265217, p(1) = 0.000734513, and so on. According to the Poisson distribution in Eq. (1.4), the probability of having at least one match (i.e., x ≥ 1) that is as good or better than the reported match, or in other words the probability of having a raw score (R) at least as large as the observed raw score (R_{obs}), is

$$pr(x \geq 1) = pr(R \geq R_{obs}) = 1 - p(0) = 1 - e^{-E} = 1 - e^{-mne^{-\lambda R}} \qquad (1.10)$$

where we have substituted E with the expression in Eq. (1.6). Note that BLAST scales the E value with a parameter K, which makes the BLAST output of the E value a bit too conservative and is perhaps not necessary.

The last term in Eq. (1.10) has a rather special term associated with it in probability theory. In short, the probability distribution

$$p(R) = e^{-mne^{-\lambda R}} \qquad (1.11)$$

with the ugly and cumbersome exponential of an exponential, is a special form of the extreme value distribution or EVD, also referred to as the Gumbel distribution in honor of the pioneer of the statistics of extremes (Gumbel, 1958). The EVD is used in BLAST (Altschul *et al.*, 1990; Altschul *et al.*, 1997) and FASTA (Pearson, 1998) to attach statistical significance to a match score between two sequences.

The E-value is the expected number of random matches with match scores that are equally good or better than the reported one. It is not a probability. This should have been obvious to anyone, yet my students often refer to it as a probability (which partially reflects how ineffective I am as a teacher). However, when E is very small, then it can be approximately interpreted as the probability of finding one match that is as good as or better than the reported one. This is shown below based on the Poisson distribution:

$$p(1) = e^{-E}E \approx E \text{ because } \lim_{E \to 0} e^{-E} = 1 \qquad (1.12)$$

An inverse problem is to obtain the length of an exact match given a critical E-value. For example, the E-value is an input parameter during a BLAST search. One can take such an input E-value to find the critical length of exact string matches, designated as L_{crit}. This L_{crit} allows us to quickly eliminate all exact string matches shorter than L_{crit}. So how to obtain L_{crit}?

A naïve approach is to try to solve the following equation for L$_{crit}$:

$$E_{crit} = mn2^{-2L_{crit}} = (L_Q - L_{crit} + 1)(L_D - L_{crit} + 1)2^{-2L_{crit}} \tag{1.13}$$

It turns out that Eq. (1.13) does not have an analytical solution for L$_{crit}$ because L$_{crit}$ is an implicit function of E$_{crit}$. To obtain L$_{crit}$ given E$_{crit}$, one would have to use numerical solution by iteration. Efficient iteration methods are available (Press *et al.*, 1992), but one may also use the approximate method below:

$$L_{crit} = L_{guess} + \frac{\ln\left(\frac{E_{guess}}{E_{crit}}\right)}{2\ln(2)} \tag{1.14}$$

where

$$E_{guess} = L_Q L_D 2^{-2L_{guess}} \tag{1.15}$$

$$L_{guess} = -\frac{\ln\left(\frac{E_{crit}}{L_Q L_D}\right)}{2\ln(2)} \tag{1.16}$$

L$_{guess}$ is derived from Eq. (1.13) by assuming L$_Q$ >> L$_{crit}$ and L$_D$ >> L$_{crit}$, so that (L$_Q$ - L$_{crit}$ +1) ≈ L$_Q$ and (L$_D$ - L$_{crit}$ +1) ≈ L$_D$. For this reason L$_{guess}$ should be constrained to be no larger than (L$_D$ – 1) or (L$_Q$ – 1).

Now if we have L$_Q$ = 100, L$_D$ = 10000, E$_{crit}$ = 0.01, then L$_{guess}$ = 13.28771, and,

$$\begin{aligned} E &= mn2^{-2L_{guess}} = 0.876 \\ L_{crit} &= L_{guess} + \ln(E/E_{crit})\ /\ \ln(4) = 13.192251 \end{aligned} \tag{1.17}$$

Thus, when BLASTing a query sequence of 100 bases against a database sequence of 10,000 bases with a critical E-value of 0.01, we may ignore those with an exact string match shorter than 13 bases. Table 1-1 shows that Eq. (1.14) is good over a wide range of L$_Q$ and L$_D$ values. I should emphasize here again that one should set sensible E$_{crit}$ for calculating L$_{crit}$. For example, if L$_Q$ = 10 and L$_D$ = 10,000,000 bases, one would be silly to compute L_{crit} by setting $E_{crit} = 0.01$ or smaller. As I have mentioned before, it is impossible to have any sequence match with $E_{crit} = 0.01$ or smaller in this case. Consequently, any L_{crit} calculated from such a E_{crit} would be of no use.

Table 1-1. Selected results computing L_{crit} by applying Eq. (1.14) with $E_{crit} = 0.01$. The values in last column is the E-value based on L_{crit} and are very close to the preset critical value of 0.01.

L_Q	L_D	L_{guess}	E_{guess}	L_{crit}	E
1000	10000	14.94868	0.009847	14.93754	0.0100
1000	1000	13.28771	0.009756	13.26988	0.0100
10000	100	13.28771	0.008760	13.19225	0.0100
1000	15	10.25827	0.003792	9.55885	0.0112
100	15	8.59730	0.004560	8.03089	0.0108
15	15	7.22882	0.003419	6.45470	0.0118
15	1000	10.25827	0.003792	9.55885	0.0112
15	10000	11.91923	0.002718	10.97942	0.0123
15	1000000	14.00000	0.007450	13.78770	0.0111

3. STRING-MATCHING ALGORITHMS IN FASTA AND BLAST

Heuristic local similarity search algarithms, which both FASTA and BLAST algorithms belong to, generally include the following three steps: (1) finding an exactly or inexactly matched string segment, (2) evaluating the statistical significance of the match, and (3) if the match is statistically significant, extending the matched string segment in both directions by dynamic programming. The second step has already been covered in the previous section, and the third step will be covered in the chapter on sequence alignment. This section will cover the FASTA and BLAST algorithms used in the first step. It is the difference in this first step that is responsible for the higher sensitivity of the FASTA algorithm than the BLAST algorithm.

3.1 FASTA

The FASTA set of programs (Pearson, 1994; Pearson and Lipman, 1988) in fact implements a number of different search algorithms, and I will illustrate only the basic ones here. The purpose is to give computer programmers sufficient details for them to implement the algorithm for homology searching in their own programs.

Suppose we wish to find the similarity between the following query sequence (Q) and the target sequence (D), with sites numbered from 0 according to the disputable convention of computation:

```
   0123456789012345678
Q: ACCGCGACCCTGACGAATA
D: ACCGCGATGACGAATA
```

The FASTA algorithm consists of three steps in achieving the heuristic local alignment. First, it uses a special form of hashing (illustrated below) to hash D for a given word length. For simplicity, we start with a word length of 1 (in which case "word" and "letter" become synonymous, i.e., a letter is a word of length 1), with the resulting hash table shown in Table 1-2. We note that word A occurs at positions 0, 6, 9, 12, 13, and 15 in D, and these numbers occupy the first row in Table 1-2. Word C occurs at positions 1, 2, 4 and 10 and the numbers occupy the second row of Table 1-2 and so on. The numbers in Table 1-2 will be referred to as H_D numbers or H_D values, where H stands for "hash" and the subscript D stands for the target (database) sequence. H_D values are needed in the second step. For a fixed word length, the computation of H_D values could in fact be done before the user has submitted any query, and consequently would belong to what is called database pre-processing.

Table 1-2. First step in the FASTA algorithm: generating a hash table of the target sequence D with a word length of 1. H_D values are the sites of the corresponding base (A, C, G and T) found in D, e.g, base A is found at site 0, 6, 9, 12, 13 and 15, respectively.

Base	*H_D*					
A	0	6	9	12	13	15
C	1	2	4	10		
G	3	5	8	11		
T	7	14				

In algorithmic terms, a hash table is made of an array of N elements each being a linked list of variable lengths. The hash table in Table 1-2 has an array of four elements (designated A, C, G, and T) each with a linked list of length 6, 4, 4, and 2, respectively.

In the second step of the FASTA algorithm, we designate the site number of Q as S_Q and compute $S_Q - H_D$ (Table 1-3). For example, nucleotide A occurs at site 0 in Q, and the differences (i.e., $S_Q - H_D$) between this 0 and the six H_D values for A in Table 1-2 (0, 6, 9, 12, 13, and 15), are consequently (0 – 0), (0 – 6), (0 – 9), (0 – 12), (0 – 13) and (0 – 15), respectively (first row of $S_Q - H_D$ values in Table 1-3). Similarly, nucleotide C occurs at site 1 in Q. Because the H_D values for C is 1, 2, 4, and 10, respectively (Table 1-2), we have (1-1) = 0, (1-2) = 01, (1-4) = -3, and (1-10) = -9. These values occupy the second row of the S_Q-H_D values in Table 1-3.

Table 1-3. Second step in the FASTA algorithm: computing the difference between the site number of the query sequence Q (S_Q) and H_D.

Q	S_Q	S_Q - H_D					
A	0	0	-6	-9	-12	-13	-15
C	1	0	-1	-3	-9		
C	2	1	0	-2	-8		
G	3	0	-2	-5	-8		
C	4	3	2	0	-6		
G	5	2	0	-3	-6		
A	6	6	0	-3	-6	-7	-9
C	7	6	5	3	-3		
C	8	7	6	4	-2		
C	9	8	7	5	-1		
T	10	3	-4				
G	11	8	6	3	0		
A	12	12	6	3	0	-1	-3
C	13	12	11	9	3		
G	14	11	9	6	3		
A	15	15	9	6	3	2	0
A	16	16	10	7	4	3	1
T	17	10	3				
A	18	18	12	9	6	5	3

The third (and the last) step is simply to compile a frequency distribution of (S_Q – H_D) values. We note that there are 10 (S_Q – H_D) values equal to 0. This means 10 matched letters between Q and D without shifting either sequence left or right. We can also find 11 (S_Q – H_D) values equal to 3 in Table 1-3, which means that by shifting D 3 positions to the right against Q, we will get a match of 11 letters:

```
Q: ACCGCGACCCTGACGAATA
D: ---ACCGCGATGACGAATA
```

For matching two long sequences, it is more informative to generate a histogram of the (S_Q – H_D) values (Figure 1-2). The histogram not only helps us visually see which value has the highest frequency, but is also very useful for obtaining nonintersecting matched segments. For example, knowing that one can get a match of 10 letters without shifting D left or right relative to Q and that one can get a match of 11 letters by shifting D 3 spaces to the right against Q helps us identify two non-intersecting matched segments as follows, with the first having 7 consecutive matches and the second having 9 consecutive matches:

```
Q: ACCGCGACCCTGACGAATA
D: ACCGCGA---TGACGAATA
```

One might wish to have a formal definition of "nonintersecting matched segments". They are matched segments that do not overlap and are often defined with reference to intersecting matched segments. Take the following two sequences for example:

```
Q: ACCGCGACCCTGACGAATA
D: ACCGCGACGACCCTGACGAATA
```

There are two intersecting matched segments below, i.e., matched segments that overlap:

```
Q: ACCGCGAC......
D: ACCGCGAC......

Q: .....GACCCTGACGAATA
D: .....GACCCTGACGAATA
```

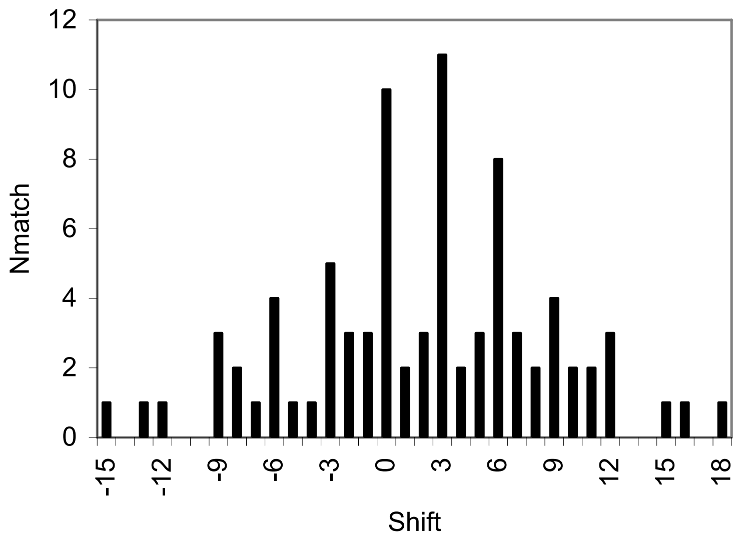


Figure 1-2. Frequency distribution of $S_Q - H_D$ values in Table 1-3.

While FASTA algorithms are for local sequence alignment, the trick it offers for finding nonintersecting segments is very important in aligning two very long sequences with sequence length in the range of millions. For such sequences, direct application of dynamic programming is often impractical because of limited memory in computers to store the matrices, e.g., with two sequences with their lengths in the order of 1,000,000, and with the scoring matrix containing only integer values each taking four bytes, the matrix alone would consume 4×10^{12} bytes. Few computers today have such a large amount of memory. However, one can use FASTA to first find nonintersecting segments and use them as anchor points and then align the sequences between anchor points by applying the dynamic programming algorithms which will be explained in the chapter on sequence alignment.

It is easy to see that, with increasing length of Q and D, $(S_Q - H_D)$ values will be too many, their frequency distribution will become less informative, and the computation will be more tedious, if we continue to use a word length of 1. In practical FASTA applications, the word length is almost always larger than 1 and would increase with the sequence length.

We now illustrate the application of the algorithm with a word length of 2. There are 16 dinucleotides and their occurrences in sequence D are listed in the first four columns of Table 1-4. These four columns are equivalent to Table 1-2 containing H_D values. The sequence Q is then represented as overlapping dinucleotides. For example, a sequence "AACG" would be represented as three overlapping dinucleotides as "AA", "AC" and "CG". Such a representation of sequence Q, together with S_Q and S_Q-H_D values, are compiled in the last six columns in Table 1-4.

Note that the hash table in Table 1-4, represented by the first four columns, is made of an array of 16 elements each with a linked list with its length varying from 0 to 3. Also note that the hash table would be mostly empty had we used a word length of 3 or 4 for comparing Q and D. The word length should increase with the sequence length of Q and D.

We have many fewer $(S_Q - H_D)$ values with a word length of 2 (Table 1-4) than with a word length of 1 (Table 1-3). This implies a substantial saving of computational time. We note eight $(S_Q - H_D)$ values equal to 3 (Table 1-4), implying eight dinucleotide matches between Q and D. These eight overlapping dinucleotides are TG, GA, AC, CG, GA, AA, AT, and TA:

```
Q: ACCGCGACCCTGACGAATA
D: ---ACCGCGATGACGAATA
```

In practical computation involving nucleotide sequences, a word length of 4 is frequently used because tetranucleotides AAAA, AAAC, …, TTTT correspond to byte values 0, 1, …, 255. In other words, each tetranucleotide can be stored in only one byte, with A, C, G, T each represented by two bits, i.e., 00, 01, 10 and 11, respectively. This means that AAAA is coded by the binary number 00000000, AAAC by 00000001, ……, and TTTT by 11111111. BLAST databases also use this encoding to save storage space for nucleotide sequences.

Table 1-4. Illustration of the FASTA algorithm with word length of 2. (1) A hash table of 16 dinucleotides (DiNuc) for D, with the numbers indicating the position of the dinucleotides in D, e.g., dinucleotide AA occurs at position 12 of D). (2) Q in overlapping dinucleotide representation and its corresponding site index (S_Q). (3) Computation of the ($S_Q - H_D$) values. For example, the first AC in Q occurs at site 0, so its $S_Q = 0$. AC occurs in D at positions 0 and 9, respectively, which are its H_D values. So the two ($S_Q - H_D$) values for the first dinucleotide AC in Q is (0 – 0) and (0 – 9), respectively.

(1)				(2)		(3)		
DiNuc	H_D			Q	S_Q	$S_Q - H_D$		
AA	12			AC	0	0	-9	
AC	0	9		CC	1	0		
AG				CG	2	0	-2	-8
AT	6	13		GC	3	0		
CA				CG	4	2	0	-6
CC	1			GA	5	0	-3	-6
CG	2	4	10	AC	6	6	-3	
CT				CC	7	6		
GA	5	8	11	CC	8	7		
GC	3			CT	9			
GG				TG	10	3		
GT				GA	11	6	3	0
TA	14			AC	12	12	3	
TC				CG	13	11	9	3
TG	7			GA	14	9	6	3
TT				AA	15	3		
				AT	16	10	3	
				TA	17	3		

3.2 BLAST

While FASTA is often found in many European data centers, BLAST is the main, and often the only, search engine in sequence database servers hosted in North America. So it is relevant for a bioinformatics student to gain some familiarity with the algorithm.

The BLAST algorithm has three steps. For BLAST servers, the first is the pre-processing of the database sequences and does not consume query time (because it is done before the user submits the query). Each database sequence of length L_D is chopped into overlapping words of a fixed length L, with word i starting at position i where i = 1, 2, ..., (L_Q-L+1). The frequently occurring words are eliminated to reduce the chance of random matching, and low-complexity words (e.g., AAAAAAA) are eliminated to reduce the bias in calculating probabilities and expected values. For example, if sequence D = “AAAAA” and sequence Q = “AAAA”, then matching the two sequences will lead to two exact matches of length 4. Our expectation, according to Eq. (1.5) and assuming equal nucleotide frequencies, is much smaller than two:

$$
\begin{aligned}
&m = L_Q - 4 + 1 = 4 - 4 + 1 = 1 \\
&n = L_D - 4 + 1 = 5 - 4 + 1 = 2 \\
&\mu = mn2^{-2L} = 1 \times 2 \times 2^{-2\times 4} = 0.0078125
\end{aligned}
\tag{1.18}
$$

The remaining words are organized into hash tables. For a powerful BLAST server, these hash tables for database sequences can be stored in memory. If the word length is 4, then a hash table will contain an array of 256 elements each with a linked list. The hash table for the following partial COI sequence from *Masturus lanceolatus* is shown in Table 1-5.

```
CGCUGAUUUUUCUCAACCAACCAUAAAGAUAUCGGCACCCUUUAUUUAGUAUUUGGUGCAUG
AGCCGGAAUAGUGGGAACGGCCUUAAGCCUGCUCAUUCGAGCGGAGCUAAGUCAACCUGGGG
CUCUCCUUGGAGACGACCAAAUUUACAAUGUCAUCGUCACAGCACAUGCAUUUGUAAUAAUU
UUCUUUAUAGUAAUACCAAUUAUGAUCGGGGGCUUUGGAAAUUGACUCAUCCCUCUUAUGAU
UGGGGCCCCUGAUAUGGCCUUUCCCCGGAUGAACAAUAUGAGCUUUUGACUAUUACCCCCCU
CUUUCCUCCUCCUCCUUGCUUCUUCAGGCGUCGAAGCAGGUGCCGGAACGGGGUGGACUGUC
UACCCUCCUUUAGCCGGAAAUUUAGCCCACGCAGGCGCCUCUGUUGACUUAACAAUCUUUUC
CCUUCAUCUGGCCGGCAUCUCCUCAAUUCUAGGGGCCAUUAACUUUAUCACAACAAUCAUUA
AUAUGAAACCACCUGCAAUUUCUCAAUACCAAACCCCCUUGUUUGUGUGAGCAGUCCUCAUC
ACGGCAGUACUUCUUCUUCUCUCGCUCCCAGUCCUUGCAGCU
```

Table 1-5. Hash table for the partial COI sequence from *Masturus lanceolatus.* Tetranucleotides not present in the sequence will have an empty linked list.

Word	Index	Linked list				
AAAA	0					
AAAC	1	501	526			
AAAG	2	24				
AAAT	3	142	224	389		
AACA	4	279	422	485		
AACC	5	14	18	115	502	527
AACG	6	77	356			
AACT	7	474				
AAGA	8	25				
AAGC	9	86	343			
AAGG	10					
…						
TTTG	254	51	174	219	292	537
TTTT	255	6	7	184	291	429

In the second step, the query sequence (Q) is chopped into words of the same length, and frequently occurring words (e.g., if A is very frequent, then a word made of all A's is also expected to be frequent and have a high chance of being encountered by random chance) as well as words of low complexity (e.g., known sequence repeats) are discarded. The remaining words are then searched against the hash table of the database sequences (Ds). For example, if the word length is set to 4, and the first tetranucleotide in Q is AAGG, then its index is 0×64+0×16+2×4+2 = 10 (Note that A, C, G, T are coded here as 0, 1, 2, 3) and we can immediately confirm its absence in D because there is no entry in the linked list indexed by 10 (Table 1-5). Similarly, if the search word is TTTG, then its index is 3×64+3×16+3×4+2 = 254, and we immediately know that it occurs at positions 51, 174, 219, 292, and 537 (These numbers, generated by computer, are 0-based, i.e., the first nucleotide is at position 0). Each match is a seed to be extended in both directions.

Third, the length of the consecutive exact matches is checked against a cutoff score. For example, one can use Eq. (1.14). Alternatively, the bit-score (S) can be used as a critical cutoff value:

$$S_{critical} = \log_2(mn) - \log_2(E) = \log_2(mn/E) \tag{1.19}$$

where m and n are effective query and database length, respectively, and the E-value is taken from user input (see the previous section for more details). If an observed S value (S_{obs}), calculated according to Eq. (1.8), is greater than $S_{critical}$, then the query is reported, otherwise it can be ignored.

FASTA is generally considered to be more sensitive than BLAST in homology detection. This may be attributed to the fact that BLAST starts with exact string matching, while FASTA starts with inexact string matching. The BLAST algorithm has problems in finding sequence similarities between extremely GC-biased and extremely AT-biased sequences, e.g.,

```
S: CCA CGA GGT AAA ATT ... ...
T: CCG CGG GGC AAG ATC ... ...
```

The two sequences are identical at the amino acid level and differ only at the third codon positions. One is AT-rich with its third codon positions all being A or T, whereas the other is GC-rich with its third codon positions all being C or G. BLAST will fail to report the sequence similarity because it does not have an exact match to start with. In contrast, FASTA will readily identify the sequence similarity. You should apply the FASTA algorithm to these two sequences as an exercise.

Although FASTA is generally more sensitive than BLAST, I should add here that both are heuristic algorithms that can find a solution that is quite good but not necessarily optimal. To guarantee the finding of the best match(es), one needs to use the sequence alignment method with dynamic programming (the subject of the next chapter).

Heuristic algorithms are used often in solving problems that are computation intensive, and are of great practical value. A well known example to illustrate the point is the problem of searching for the largest corn in a large cornfield versus the problem of searching for a "very large" corn, say within the top 1%. The first involves the measuring and comparison of all corns in the cornfield, which may be extremely laborious, although you are guaranteed to find the largest corn. In contrast, the second problem can be easily solved by taking a random sample of corns, fitting a statistical distribution to the corns, finding the size of the corn that lies above 99% of the distribution, and using this criterion to search for the corn that is larger than the fixed criterion (or simpler, if the sample is sufficiently large, picking the largest corn in the sample). In addition, after estimating the density of the corn and measuring the size of the cornfield, we can estimate the total number of corns in the field and obtain a fairly good estimate of the size of the largest corn. The formulation and solution of the second problem belong to the heuristic approach.

The heuristic approach is used not only in biology or science, but also in sociology, psychology and economics. For example, Herbert A. Simon, Nobel laureate in economics in 1978, contributed significantly to the revelation of practical human decision-making as a process of searching for satisfactory solutions by heuristic methods rather than optimal solutions by optimality models. Choosing a religion to guide our behavior is in most cases based on the result of a heuristic approach. In our life time, we will never be able to do an exhaustive comparison of different religions, and we typically adopt one that seems to work pretty well for us and for our families. The happy ending in the movie "Pretty Woman" is a good example of a successful application of a fast heuristic approach. Indeed, one may never get married if one is determined to find the best spouse. However, a student of mine has brought my attention to the increasing divorce rate which serves as a good illustration of the failure of the heuristic approach in building a family. This highlights an important point concerning heuristic method. Being heuristic does not mean that one would settle with any solution, even a very haphazard one. So don't take me responsible for your hasty heuristic approach in real life.

4. HOMOLOGY SEARCH AND SEQUENCE ANNOTATION

Both genomic sequencing and large-scale characterization of expressed sequence tags (ESTs) demand efficient computational tools for automatically annotating the large number of the resulting sequence reads. I have already mentioned at the beginning of the chapter that two major categories of gene-annotation methods are in current use, with one based on known genes in molecular databases, and the other based on known gene structures.

I wish to highlight two important points here. First, the rapid increase of well-annotated genomic databases coupled with the improvement of the special-purpose databases for protein functional classification such as COG (Tatusov *et al.*, 2003; Tatusov *et al.*, 1997), pFAM (Bateman *et al.*, 1999; Bateman *et al.*, 2004), SMART (Letunic *et al.*, 2004; Letunic *et al.*, 2002; Ponting *et al.*, 1999; Schultz *et al.*, 2000), pSort (Nakai and Horton, 1999) and CDD (Marchler-Bauer *et al.*, 2005; Marchler-Bauer *et al.*, 2002) has dramatically increased the popularity of the first approach. A number of EST annotation platforms using the first approach are now available (Ayoubi *et al.*, 2002; Davila *et al.*, 2005; Koski *et al.*, 2005; Mao *et al.*, 2003; Martin *et al.*, 2004; Paquola *et al.*, 2003), based on searching against the non-redundant GenBank files or/and special-purpose databases for protein functional classification.

Second, the proliferation of the primary databases and of the secondary sequence annotation platforms have given rise to two major problems for practicing researchers (Marchler-Bauer *et al.*, 2005; Marchler-Bauer *et al.*, 2002). The first is that any large-scale query against any of these primary databases such as COG (Tatusov *et al.*, 2003; Tatusov *et al.*, 1997), pFAM (Bateman *et al.*, 1999; Bateman *et al.*, 2004), SMART (Letunic *et al.*, 2004; Letunic *et al.*, 2002; Ponting *et al.*, 1999; Schultz *et al.*, 2000) will take unbearable amount of time. To make things worse, different databases contain overlapping but not identical subsets of proteins and protein families, and one often has to query these databases sequentially in order to maximize the chance of having a good hit. Second, different databases have different methods for classifying proteins into functional families and a sequence query against different databases may yield conflicting results. CDD (Marchler-Bauer *et al.*, 2005; Marchler-Bauer *et al.*, 2002) was created mainly in response to these two problems. First, it imports and cross-validates the protein annotations from major protein function databases such as COG (Tatusov *et al.*, 2003; Tatusov *et al.*, 1997), pFAM (Bateman *et al.*, 1999; Bateman *et al.*, 2004), SMART (Letunic *et al.*, 2004; Letunic *et al.*, 2002; Ponting *et al.*, 1999; Schultz *et al.*, 2000), removes redundant annotations, resolves annotation conflicts and augment the database entries by adding other curated protein sequences. Second, CDD uses the RPS-BLAST search engine to dramatically increase the search speed. This is augmented by pre-computation of much of the output. The joint effect of these improvements results in more hits, fewer conflicts and shorter search time than before. One particular advantage of CDD is its web API (**A**pplication **P**rogramming **I**nterface) which allows programmers to perform automated searches and annotations.

5. POSTSCRIPT

A student of mine once told me that she always found it miraculous to receive many homologous sequences from diverse organisms after BLASTing a human gene against GenBank. The various forms of nature's creation are so intricately and closely related to each other, and people of different races or different nations are so similar to each other genetically, that she found it a mystery that modern humans were still so ready and willing to kill and torture each other. "Aren't we killing ourselves by killing people with their genomes essentially identical to ourselves?" she asked.

I think we are, and we should stop. I can't resist the temptation of concluding this chapter with a quote from Albert Einstein (Einstein *et al.*, 1931, p. 6) when he was discussing man's relationship to others: "This subject brings me to that vilest offspring of the herd mind - the odious militia. The man who enjoys marching in line and file to the strains of music falls below my contempt; he received his great brain by mistake - the spinal cord would have been amply sufficient. This heroism at command, this senseless violence, this accursed bombast of patriotism - how intensely I despise them! War is low and despicable, and I had rather be smitten to shreds than participate in such doings."

Yet in spite of Einstein's antiwar stance, his successes have often been depicted with military analogies, such as how the established castles of physics came tumbling down at the trumpet of his theory of relativity, as if Einstein is a military general bent on conquering. In a similar vein, Louis Pasteur was accredited with military and strategic genius, that "he had something of Napoleon in his way of always taking the initiative, of suddenly changing the terrain, of showing up where he was least expected, of suddenly concentrating his forces in a narrow sector to make the breakthrough, …… Without a doubt, Pasteur's saga was as stirring as Napoleon's!" (Jacob, 1988, p. 248).

Why does a life of saving have to be glorified with a life of killing?

Why should human evolution in the 21st century still be shaped by the ugly selection force called wars?

Einstein signed his last letter, one week before his death, giving permission to have his name on a manifesto urging all nations to give up nuclear weapons. May the international peace he had dreamed of all his life arrive sooner!

Chapter 2

SEQUENCE ALIGNMENT

1. INTRODUCTION

Sequence alignment is not only the essential first step in molecular phylogenetics, quantification of substitution patterns, and dating of speciation and gene duplication events, but also a powerful tool for identify mutations leading to genetic diseases. For example, aligning the β-hemoglobin gene sequence from one type of β-thalassemia against the normal β-hemoglobin gene immediately reveals an insertion of T at site 79 (Figure 2-1).

```
                10        20        30        40        50        60
        ----|----|----|----|----|----|----|----|----|----|----|----|--
Normal  ATGGTGCACCTGACTCCTGAGGAGAAGTCTGCCGTTACTGCCCTGTGGGGCAAGGTGAACGT
Thalas. ATGGTGCACCTGACTCCTGAGGAGAAGTCTGCCGTTACTGCCCTGTGGGGCAAGGTGAACGT
        ************************************************************
              70        80        90       100       110       120
        --|----|----|----|----|----|----|----|----|----|----|----|----
Normal  GGATGAAGTTGGTGGT-GAGGCCCTGGGCAGGTTGGTATCAAGGTTACAAGACAGG......
Thalas. GGATGAAGTTGGTGGTTGAGGCCCTGGGCAGGTTGGTATCAAGGTTACAAGACAGG......
        **************** ***************************************
```

Figure 2-1. Alignment between the normal and mutant β-hemoglobin gene sequences.

The insertion creates an inframe stop codon and results in a truncated β-hemoglobin protein. When the β-hemoglobin locus is heterozygous with one mutant and one normal gene, the carrier is said to have β-thalassemia minor and can be easily detected by gel electrophoresis because the mutant β-hemoglobin, being much shorter than the normal, would migrate much faster on the gel under an electric field.

This chapter covers (1) pairwise global and local alignment by dynamic programming with different scoring schemes, from the simplest scoring scheme with match/mismatch scores and gap penalties all specified by constants, to more useful scoring scheme with match/mismatch scores specified by a similarity matrix and gap penalties specified by the affine function, (2) profile alignment between one sequence and a set of aligned sequences which is essential for practical implementation of multiple sequence alignment, and (3) multiple alignment that is reduced to pair-wise alignment and profile alignment by using a guide tree. Most textbooks on bioinformatics omit the affine function, and no textbook I know of includes any detailed explanation of profile alignment.

Dynamic programming algorithms constitute a general class of algorithms not only used in sequence alignment, but also in many other applications. For example, the Viterbi algorithm and the forward algorithm used in hidden Markov models (HMM) are also dynamic programming algorithms. We will cover HMM in great detail and illustrate with numerical examples latter. Learning the dynamic programming algorithms used in sequence alignment paves the way for more advanced applications in latter chapters.

Sequence alignment methods, especially those for obtaining multiple alignments, are central to molecular biology, evolution and phylogenetics. One of the global sequence alignment programs, ClustalW (Higgins and Sharp, 1988; Thompson *et al.*, 1994) is probably the second most used bioinformatics tool next to the BLAST suite of programs (Altschul *et al.*, 1990; Altschul *et al.*, 1997).

2. PAIRWISE ALIGNMENT

Given two strings S ($=s_1s_2...s_n$) and T ($=t_1t_2...t_m$), a pairwise alignment of S and T is defined as an ordered set of pairings of (s_i, t_j) and of gaps (s_i,−) and (−,t_j), with the constraint that the alignment is reduced to the two original strings when all gaps in the alignment are deleted. A prefix of S, specified here as S_i, is a substring of S equal to $s_1s_2...s_i$, where $i \leq n$.

An optimal alignment is operationally defined as the pairwise alignment with the highest alignment score for a given scoring scheme. For this reason, an optimal alignment is meaningless without the specification of the scoring scheme.

Alignment by dynamic programming guarantees that the resulting alignment is the optimal alignment or one of the equally optimal alignments. We will first illustrate the global pairwise alignment (Needleman and Wunsch, 1970) followed by local pairwise alignment (Smith and Waterman, 1981b). Local sequence alignment is for searching local similarities between sequences, e.g., homeobox genes which are not similar globally but all share a very similar homeodomain motif. The best known algorithm for local alignment is Smith and Waterman (1981b).

Here we will first learn a simple dynamic programming algorithm for pairwise alignment using a simple scoring scheme with constant gap penalty. This is then extended in two ways, first by introducing a similarity matrix to replace match and mismatch scores, and second by introducing the affine function to better approximate the origin of the insertion and deletion during sequence evolution.

2.1 Pairwise alignment with constant gap penalty

2.1.1 Global alignment

Suppose we want to align two sequences S and T with S = ACGT and T = ACGGCT. A simple scoring scheme is used with a constant gap penalty (G) of -2, a match score (M) of 2 and a mismatch score (MM = -1). Global alignment with the dynamic programming approach is illustrated numerically in Figure 2-2. One sequence of the two sequences occupies the top row and will be referred to hereafter as the row sequence (sequence S in our example). The other sequence occupies the first column and will be referred hereafter as the column sequence (sequence T in our example). Based on these two sequences, two matrices are computed. The first is the scoring matrix to obtain the alignment score, with the dimensions (n+1, m+1). The second is the backtrack matrix needed to obtain the actual alignment, with the dimensions (n,m). In Figure 2-2, the two matrices are superimposed, with the scoring matrix being the numbers and the backtrack matrix being made of arrows. The backtrack matrix is sometimes called the traceback matrix. However, the word traceback is marked by an annoying red wavy line in Microsoft WORD. So my choice of the two is obvious.

A value in row i and column j in the scoring matrix is the alignment score between a prefix of S and a prefix of T, i.e., S_j and T_i. This will become clear later.

The first row and the first column of the scoring matrix is filled with i×G (where i = 0, 1, ..., n) and j×G (where j = 0, 1, ..., m), respectively. They represent consecutive insertion of gaps. For example, the number -8 in the last cell of the first row of the scoring matrix implies the following alignment with four consecutive gaps in the column sequence and an alignment score of -8:

```
ACGT
----
```

		A	C	G	T
	0	-2	-4	-6	-8
A	-2	↖ 2	← 0	← -2	← -4
C	-4	↑ 0	↖ 4	← 2	← 0
G	-6	↑ -2	↑ 2	↖ 6	← 4
G	-8	↑ -4	↑ 0	↖↑ 4	← 5
C	-10	↑ -6	↑ -2	↑ 2	↖ 3
T	-12	↑ -8	↑ -4	↑ 0	↖ 4

Figure 2-2. . Computation involved in obtaining the scoring and the backtrack matrices (superimposed) with the match score equal to 2, mismatch -1 and the gap penalty equal to -2. There are two equally optimal alignments each with an alignment score of 4.

Similarly, the number of -12 in the last cell of the first column of the scoring matrix implies the following alignment with six consecutive gaps on the row sequence and an alignment score of -12:

```
------
ACGGCT
```

The first cell where we need to compute the score is the one corresponding to the first character of S and T, i.e., the cell with a value of 2. To compute the value for the cell, we need values in three other cells, one to its left, one above it and one to its upleft, with their cell values designated as L, U and UL, respectively. Note that the cell has an upleft (UL) value of 0, a top value (U) equal to -2, and a left value (L) equal to -2. The following three values are calculated:

DIAG = UL + IF(Corresponding characters match, M, MM) = 0 + 2 = 2
LEFT = L + G = -2 + (-2) = -4
UP = U + G = -2 + (-2) = -4

The IF function above takes the value of M if the two corresponding nucleotides match, or MM if they do not. The maximum of these three values is DIAG, i.e., 2, which was entered as the first computed element in the scoring matrix. The cell is also filled with an upleft arrow because DIAG is the maximum of the three values. If LEFT (or UP) happened to be the maximum of the three, we would have put a left-pointing (or up-pointing) arrow in the corresponding cell in the backtrack matrix.

The computation is from left to right and from top to bottom. For the second cell, the maximum of the three values is LEFT (= 0), and the corresponding cell in the backtrack matrix is filled with a left-pointing arrow. We continue the computation to the bottom right cell, with the final value in the bottom-right value equal to 4. This is the alignment score. You may note that the cell corresponding to the nucleotide G in the row sequence and the second G in the column sequence is special with two arrows. You will find that the DIAG and UP values are both equal to 4 in this cell. Hence both the upleft and the up-pointing arrows in this cell. Such a cell implies the existence of equally optimal alignments.

The aligned sequences are obtained directly from the backtrack matrix. We start from the bottom-right cell and follow the direction of the arrow in the cell. The upleft arrow in the bottom-right cell means that we should stack the two corresponding nucleotides (T and T) in the row and column sequences (Figure 2-3). Note that you would be stacking the two corresponding nucleotides regardless of whether they are the same or different as long as an upleft arrow is in the cell. A left-pointing or up-pointing arrow in the cell means a gap in the column sequence or row sequence, respectively.

```
(a)            (b)

654321         654321
ACG--T         AC-G-T
ACGGCT         ACGGCT
```

Figure 2-3. The protocol of obtaining the sequence alignment by following the backtrack matrix. The numbers in the first row show the order of obtaining the alignment site by site from the last to the first (i.e., backtracking).

The upleft arrow in the bottom-right cell leads us to the cell containing an up-pointing arrow, meaning a gap in the row sequence, i.e., we stack a gap character “-” over the corresponding nucleotide (C) in the column sequence (Figure 2-3). This up-pointing arrow brings us to the special cell with two arrows, one pointing upleft and the other up (Figure 2-2). This leads to alternative construction of the sequence alignment. If we choose the up-pointing arrow, we will stack a gap character over the corresponding nucleotide (G) in the column sequence and proceed to the cell with a value of 6 and an upleft arrow. This ultimately leads to the sequence alignment in Figure 2-3a. Alternatively, we may choose to follow the upleft arrow and stack the two nucleotides (G in both sequences) as shown in Figure 2-3b. This ultimately leads to the alternative sequence alignment in Figure 2-3b.

Both alignments in Figure 2-3 have four matches, two gaps, and zero mismatch. So the alignment score is 4×M+2×G+0× (MM) = 4, which we already know after completing the scoring matrix whose bottom-right cell contains the alignment score.

Recall that each cell with a score and an arrow specifies an optimal alignment between a prefix of S and a prefix of T, i.e., S_j and T_i. For example, the first cell with a calculated value of 2 and an upleft arrow specifies the optimal alignment of S_1 (= ‘A’) and T_1 (= ‘A’) with an alignment score of 2 (which is the match score). The next cell to the right, with a score of 0 and a left arrow, specifies the optimal alignment of S_2 and T_1 (The 0 score results from a match and a gap penalty):

```
AC
A-
```

The cell with a score of 4 and two arrows specifies two equally optimal alignments both with an alignment score of 4:

Alignment 1:
```
ACG-
ACGG
```

Alignment 2:
```
AC-G
ACGG
```

When sequences are long, there might be many equally optimal alignments and few computer programs would try to find and output all of them. Instead, only one path is followed, leading to the output of only one of the potentially many equally optimal alignments.

Dynamic programming guarantees that the resulting alignment is optimal given the scoring scheme. In other words, there is no alignment that can have an alignment score greater than 4 given the two sequences and the scoring scheme of M = 2, MM = -1 and G = -2. However, an optimal alignment may change when the scoring scheme is changed. This is illustrated in Figure 2-4, where the use of scoring scheme 1 would result in Alignment 1 (with alignment score = 14) better than Alignment 2 (with alignment score = 12) but the use of scoring scheme 2 would lead to the opposite, with Alignment 2 (alignment score = 20) much better than Alignment 1 (alignment score = 9).

```
Alignment 1: ACCCAGGGCTTA
             ACCCGGGCTTAG

Alignment 2: ACCCAGGGCTTA-
             ACCC-GGGCTTAG

Scoring scheme 1: M = 2, MM = 0, G = -5
Scoring scheme 2: M = 2, MM = -1, G = -1
```

Alignment	Match	Mismatch	Gap	Score1	Score2
1	7	5	0	14	9
2	11	0	2	12	20

Figure 2-4. Illustration of the dependence of optimal alignment on scoring scheme. Score1 and Score2 in the bottom table refer to Scoring scheme 1 and Scoring scheme 2, respectively.

Because there is no objective way of choosing the right scoring scheme, it is therefore important to keep in mind that sequence alignment is a method of data exploration instead of an analytical method that will lead to a single best solution. For this reason, nearly all computer programs for sequence alignment allow the user to try various scoring schemes and post-alignment manual editing.

2.1.2 Local alignment

Local sequence alignment (Smith and Waterman, 1981b) is similar to global alignment presented above, with only three major differences. First, the first row and the first column of the scoring matrix is filled with zero instead of i×G. Second, whenever the cell value becomes negative (i.e., the maximum of the three values is smaller than 0), the cell value is set to 0. Thus, when two sequences have a short but perfect local match, and little similarity elsewhere, the alignment score of the short but perfect match is not affected by the low similarity elsewhere. Third, because a local alignment can end anywhere in the matrix, we will not trace back from the cell in the bottom-right corner of the score matrix. Instead, we find the maximal score in the matrix and trace back from that point until we reach a cell with a value of 0, which indicates the start of the local alignment.

In Chapter 1 we have already covered two widely used heuristic methods for local alignment, i.e., BLAST and FASTA. In Chapter 7 we will learn Gibbs sampler which is another method for searching local similarities and local alignment.

2.1.3 The simple scoring scheme needs extension

The simple scoring scheme that we have used has three major problems. First, transitions (i.e., substitutions between nucleotides A and G and between C and T) generally occur more frequently than transversions (When A or G is replaced by C or T). This suggests that we should not treat transitional differences and transversional differences with the same mismatch score. Instead, transitions should be penalized less than transversions. Second, there are often ambiguous bases in input the sequences, e.g., R for A or G and Y for C or T. An A-R pair is neither a strict match nor a strict mismatch, but has a probability of 0.5 being a match and a probability of 0.5 being a transition. The simple scoring scheme we have used cannot handle these problems, which necessitates the use of a similarity matrix.

The simple scoring scheme also has another, perhaps even more serious problem, caused by constant gap penalty. A biologist will complain loudly that an alignment method is wrong if it considers the two alternative alignments in Figure 2-3 as equally good, and would have chosen the alignment in Figure 2-3a as a better alignment. The simplest solution to this problem is to use what is called an affine function for gaps. In the following sections, we will learn these two extensions, first with a similarity matrix and second with an affine function.

2.2 Pairwise alignment with a similarity matrix

2.2.1 DNA matrices

One example of a similarity matrix is the “transition bias matrix” (Table 2-1) used in DAMBE (Xia, 2001; Xia and Xie, 2001b) for multiple alignment with a star tree. A start tree contains only one internal node with all leaves connected to this same internal node. The meaning of the top 4×4 matrix (bolded values in Table 2-1) is easy to understand. The first four diagonal values of 30 are equivalent to the match score, a mismatch score involving a transversion or a transition is -30 or 0, respectively, because transitions in general occur much more frequently than transversions and consequently penalized less (Table 2-1). The rest of the matrix (Table 2-1) involves ambiguous codes specified in Table 2-2, according to the Nomenclature Committee of the International Union of Biochemistry (1985).

Table 2-1. A similarity matrix accommodating the transition bias frequently observed in nucleotide substitutions.

A	C	G	U	R	Y	M	W	S	K	D	H	V	B	N
30														
-30	**30**													
0	**-30**	**30**												
-30	**0**	**-30**	**30**											
15	-30	15	-30	15										
-30	15	-30	15	-30	15									
0	0	-15	-15	-8	-8	0								
0	-15	-15	0	-8	-8	-8	0							
-15	0	0	-15	-8	-8	-8	-15	0						
-15	-15	0	0	-8	-8	-15	-8	-8	0					
0	-20	0	-10	0	-15	-10	-5	-10	-5	-3				
-10	0	-20	0	-15	0	-5	-5	-10	-10	-10	-3			
0	-10	0	-20	0	-15	-5	-10	-5	-10	-7	-10	-3		
-20	0	-10	0	-15	0	-10	-10	-5	-5	-10	-7	-10	-3	
-8	-8	-8	-8	-8	-8	-8	-8	-8	-8	-8	-8	-8	-8	-8

Table 2-2. IUB codes of nucleotides.

Code	Meaning	Complement
A	A	T
C	C	G
G	G	C
T/U	T	A
M	A or C	K
R	A or G	Y
W	A or T	W
S	C or G	S
Y	C or T	R
K	G or T	M
V	A or C or G	B
H	A or C or T	D
D	A or G or T	H
B	C or G or T	V
X/N	G or A or T or C	X
-	Gap (not G or A or T or C)	-

The coding scheme is often refereed to as the IUB code or IUB notation. For example, R represents either A or G, so an A-R pair has a probability of 0.5 being an A-A match and a probability of 0.5 being an A-G transition. The corresponding score (=15 in Table 2-1) is consequently somewhere between a perfect match and a transition. In contrast, Y stands for either C or T/U and an A-Y pair is always a transversion, with a score of -30 (Table 2-1).

2.2.2 Protein matrix

Amino acids differ from each other in volume, charge, polarity and many other properties (Figure 2-5), and amino acid residues in a protein confer to the protein different properties. Proteins with long half-life (>1 day) typically have glycine, valine or methionine at their N-terminus, whereas those with short half-life (a few minutes) typically have positively charged residues (arginine, lysine) at their N-terminus. A small amino acid residue such as glycine and alanine at the penultimate site (the second amino acid site in the nascent peptide) allows the initiator methionine to be efficiently cleaved (Moerschell *et al.*, 1990). Amino acid replacements involving very different amino acids are generally selected against (Xia and Li, 1998). For this reason, a scoring scheme with only match and mismatch is rarely used for protein sequence alignment.

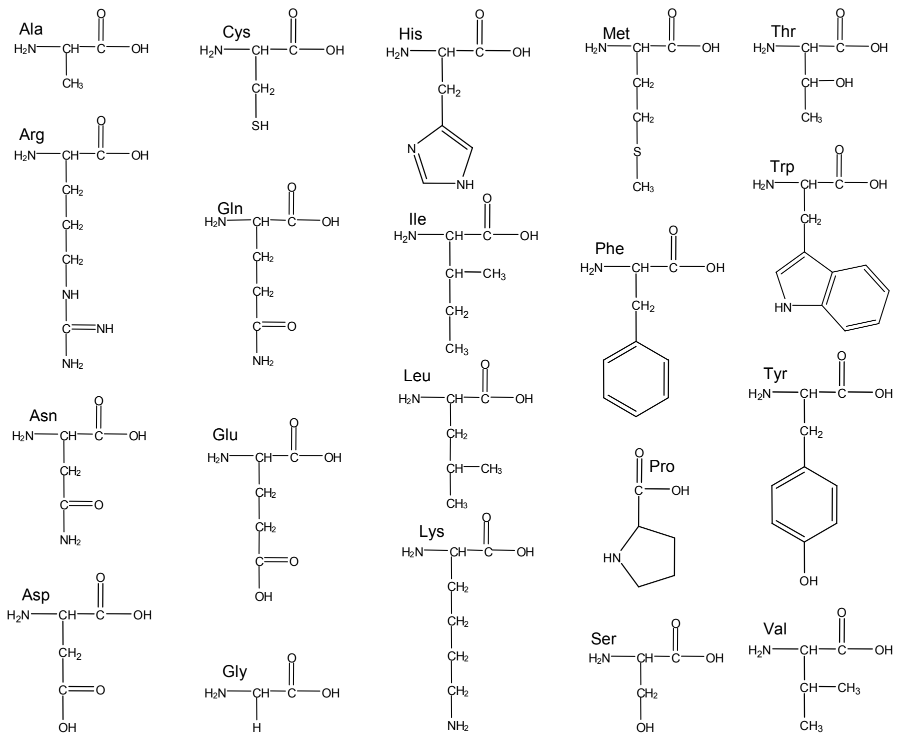
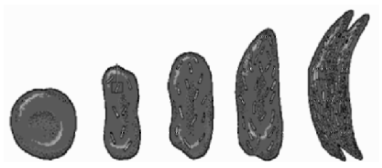


Figure 2-5. Structural formula of 20 amino acids.

A frequently used example to illustrate the effect of an amino acid being replaced by a different amino acid is the sickle-cell anemia. Sickle-cell anemia is caused by a single amino acid replacement in the β-chain of the human hemoglobin at the six position, with a glutamate residue replaced by a valine residue (Figure 2-6). Glutamate is negatively charged and hydrophilic, and tends to stay on the surface of the protein in the aqueous environment in the blood. In contrast, valine is a non-polar and hydrophobic residue and tends to shrink into the middle of the protein. The deformed protein molecules then form bundles and distort the red blood cell that carries them, resulting in the characteristic shape of a sickle (Figure. 2-6). It is generally true that amino acids of different polarity rarely replace each other (Xia and Li, 1998), whereas amino acids with similar polarity can replace each other quite frequently (Xia and Kumar, 2006).



```
Hb-A: Val-His-Leu-Thr-Pro-Glu-Glu……
Hb-S: Val-His-Leu-Thr-Pro-Val-Glu……
```

Figure 2-6. Sickle-cell anemia is caused by a single amino acid replace of a glutamate residue at the sixth position (Hb-A allele) by a valine residue (Hb-S allele). The mutant deformed hemoglobin molecules distort the red blood cell which progresses from the normal disk-like shape to the sickle-like shape.

Frequently used substitution matrices for protein sequences are of two types, the PAM matrix (Dayhoff *et al.*, 1978) and the BLOSUM matrix (Henikoff and Henikoff, 1992). The letter codes for amino acids proposed by the Nomenclature Committee of the International Union of Biochemistry (1985) are shown in Table 2-3. These codes are now universally adopted by the scientific community. I have omitted an introduction of these matrices because (1) the limit of page size of the book precludes the presentation of 20×20 matrices and (2) an excellent introduction of these matrices appropriate for readers of this book is already available (Higgs and Attwood, 2004). In short, both PAM and BLOSUM matrices are derived from sequence alignment related proteins, with the former based on global alignment and the latter based on local alignment. The PAM1 matrix is based on comparisons of sequences with no more than 1% divergence and all other PAM matrices are derived from this PAM1 matrix. The requirement of proteins with no more than 1% divergence is necessary for reliable global alignment. The most frequently used BLOSUM matrix is BLOSUM 62 which is calculated from comparisons of sequences with no less than 62% divergence. BLOSUM xx matrix is based on sequence blocks with no less than xx% divergence, i.e., all BLOSUM matrices are based on observed alignments in contrast to the PAM matrices all derived from the PAM1 matrix. BLOSUM 62 is the default matrix in BLAST 2.0.

Table 2-3. IUB letter codes of amino acids.

1-letter	3-letter	Meaning	Codon[1]
A	Ala	Alanine	GCT,GCC,GCA,GCG
B		Asp or Asn	GAT,GAC,AAT,AAC
C	Cys	Cysteine	TGT,TGC
D	Asp	Aspartic	GAT,GAC
E	Glu	Glutamic	GAA,GAG
F	Phe	Phenylalanine	TTT,TTC
G	Gly	Glycine	GGT,GGC,GGA,GGG
H	His	Histidine	CAT,CAC
I	Ile	Isoleucine	ATT,ATC,ATA
K	Lys	Lysine	AAA,AAG
L	Leu	Leucine	TTG,TTA,CTT,CTC,CTA,CTG
M	Met	Methionine	ATG
N	Asn	Asparagine	AAT,AAC
P	Pro	Proline	CCT,CCC,CCA,CCG
Q	Gln	Glutamine	CAA,CAG
R	Arg	Arginine	CGT,CGC,CGA,CGG,AGA,AGG
S	Ser	Serine	TCT,TCC,TCA,TCG,AGT,AGC
T	Thr	Threonine	ACT,ACC,ACA,ACG
V	Val	Valine	GTT,GTC,GTA,GTG
W	Trp	Tryptophan	TGG
X	Xxx	Unknown	
Y	Tyr	Tyrosine	TAT,TAC
Z		Glu or Gln	GAA,GAG,CAA,CAG
*	End	Terminator	TAA,TAG,TGA

(1) assuming the standard genetic code.

2.3 Pairwise alignment with gap penalty specified by the affine function

The second extension of the simple scoring scheme is to replace the constant gap penalty with what is called an affine function. The problem with the constant gap penalty is exemplified in the two optimal alignments in Figure 2-3. From a biological point of view, the alignment with two independent gaps (Figure 2-3b) is less likely than the one with only one gap of length 2 (Figure 2-3a). So we should find a gap penalty scheme that favors the alignment in Figure 2-3a against the one in Figure 2-3b. The affine function, which is used in BLAST (Altschul *et al.*, 1990; Altschul *et al.*, 1997), is the simplest of the gap penalty schemes that will do the job. One particular advantage of the affine function is that it allows the alignment to be completed in time proportional to MN, where M and N are the length of the two sequences to be aligned.

The affine function for gap penalty is specified as

$$G(x) = -(a + bx) \tag{2.1}$$

where x is the length of the gap, and a and b are the gap open and gap extension penalties, respectively. The gap penalty increases linearly with the length of the gap. BLAST has its defaults with a = 5 and b = 2, together with the match score (M) = 1 and mismatch score (MM) = -3.

Figure 2-7 illustrates the computation involved in aligning two sequences with M = 1, MM = -3 and the gap penalty specified with a = 5 and b = 2, i.e., the default BLAST scoring scheme. Note that, while the first value in the matrix is still 0 (Figure 2-7) as before, the next value on the first row or first column is -7 which results in -(a + 1b) = -(5 + 1 × 2) = -7 (Note that a shift leftward or downward means inserting a gap either in the row or in the column sequence, respectively, and the first gap is associated with both the gap open and gap extension penalties. The values after -7 on the first row or on the first column are decreased by gap extension only, i.e., if a gap is already open, additional gaps will only suffer from gap extension penalties.

We again need to calculate three values in each remaining cells. In general, we calculate the DIAG, LEFT and UP values as specified below and fill the cell with the maximum of the three,

DIAG = UL + if(match, M, MM)
LEFT = L – If(GapOpened already, 0, a) – b
UP = U – if(GapOpened already, 0, a) – b

For the first cell, DIAG = 1 because of the match of the two corresponding nucleotides, i.e., the A-A pair. The UP and LEFT values are both -9. So we have 1 in the cell with an upleft arrow (Figure 2-7). For the next cell, we have

DIAG = -7 -3 = -10
UP = -9 - 2 = -11
LEFT = 1 - 5 - 2 = -6

		A	C	G	T
	0	-7	-9	-11	-13
A	-7	↖ 1	← -6	← -8	← -10
C	-9	↑ -6	↖ 2	← -5	← -7
G	-11	↑ -8	↑ -5	↖ 3	← -4
G	-13	↑ -10	↑ -7	↖↑ -4	↖ 0
C	-15	↑ -12	↑ -9	↑ -6	↖↑ -7
T	-17	↑ -14	↑ -11	↑ -8	↖ -5

Figure 2-7. Pairwise alignment with M = 1, MM = -3, and gap penalty defined by an affine function with a = 5 and b = 2.

Note that LEFT value for this cell is penalized with both the gap open and gap extension penalty because the proceeding cell (with value = 1) has an upleft arrow, i.e., no gap (Figure 2-7). If the proceeding cell had a left-pointing arrow, which means that the gap has already been opened, only the gap extension penalty would be applied. The largest value of the three is LEFT (= -6), and the cell is therefore filled with -6 with a left-pointing arrow. This process continues until we get to the last cell, with a value of -5. This is the alignment score based on the scoring scheme with gap penalties specified with the affine function.

There are a few cells that need some explanation. The first is the cell with a value of -4 corresponding to the nucleotide G in the row sequence and the second nucleotide G in the column sequence. The cell has two arrows, one pointing up and one pointing upleft (Figure 2-7). This is because both the DIAG and UP values are equal to -4:

DIAG = -5 + 1 = -4

UP = 3 – (5 + 2) = -4

Had this cell been the last cell, i.e., if we were aligning the partial row sequence of "ACG" against the partial column sequence "ACGG", we would get two alternative optimal alignments both with alignment score of -4:

```
Alignment 1: ACG-
             ACGG

Alignment 2: AC-G
             ACGG
```

The other cell with two arrows is in the last column, second row from the bottom. Had this cell been the last cell, i.e., if we were aligning the row sequence of "ACGT" against the partial column sequence "ACGGC", we would get two alternative optimal alignments both with alignment score of -7 (i.e., three matches, one mismatch and one gap open and one gap extension penalty):

```
Alignment 1: ACGT-
             ACGGC

Alignment 2: ACG-T
             ACGGC
```

The trickiest cell is at the second last column and second last row, i.e., the one with an UP value of -6 and an up-pointing arrow. The DIAG and LEFT values for this cell are simple:

DIAG = -7 – 3 = -10
LEFT = -9 – 2 = -11

However, the UP value depends on which of the arrows in the cell above (i.e., the cell with a value of -4 and two arrows) we should take. If we take the up-pointing arrow (i.e., a gap has already been opened), then UP = -4 – 2 = -6. However, if we take the upleft arrow (i.e., no gap opened yet), then UP = -4 – (5 + 2) = -11. We should choose the maximum value, i.e., -6, and this constrains the previous cell (i.e., the cell with a value of -4 and two arrows) to have an up-pointing arrow. In other words, after we have put a -6 value into the cell, the cell above will no longer have two arrows but will only have an up-pointing arrow. It is for this reason that I have set the upleft arrow in that cell to grey but left the up-pointing arrow black.

Now following the backtrack arrows, we obtain the alignment in Figure 2-3a. The alignment has four matches and one gap of length 2. So the alignment score is $4 \times 1 - (5 + 2 \times 2) = -5$, which confirms that our scoring matrix (Figure 2-7) has been obtained correctly (note that the lower right value in the scoring matrix is the alignment score).

3. MULTIPLE SEQUENCE ALIGNMENT

3.1 Profile alignment

Profile alignment aligns one sequence (designated T) against a set of already aligned sequences in the form of a profile (designated S), or align two profiles S_1 and S_2. It is an essential technique for multiple sequence alignment. There are various approaches to profile alignment. The simplest is to get a consensus sequence from S (designated C_S) and align T and C_S by using the pairwise alignment method we learned in previous sections. Whenever we insert a gap in C_S, we insert a corresponding gap in all sequences in S. However, we will learn a mathematically more acceptable approach in this section.

Suppose we want to align sequence T = "ACG" against S containing the following three aligned sequences:

```
AC-GT
AC-GT
GCCAT
```

The first step in profile alignment is to represent S with a site-specific frequency profile. The set of three aligned sequences have five symbols (A, C, G, T and the gap symbol "-") and can be represented by the profile shown in the first five rows in Figure 2-8. The first column is a list of the five symbols, followed by five data columns corresponding to five aligned sites. The first data column represents the frequencies of symbols in the first aligned site, with the frequencies of A and G being 2/3 and 1/3, respectively. The second data column represents the frequencies of the second aligned site with the frequency C being 1 and the frequencies of other symbols being 0, and so on. Thus, S can always be represented by a site-specific profile in the form of a N×L matrix where N is the number of symbols and L is the sequence length. It is important to note that any phylogenetic information among sequences in S is lost in converting the set of aligned sequences to a profile.

The profile representation in the Clustal family of programs (Higgins and Sharp, 1988; Thompson *et al.*, 1994) uses all ambiguous codes (Tables 2-2 and 2-3). In addition, two synonymous pairs of ambiguous codes are used in Clustal, the X/N pair and the T/U pair. It would be computationally more efficient to pre-processing the sequences to use either X or N (or either T or U) but not both, especially when one uses a programming language that does not support pointers, e.g., Visual Basic or Java.

	A	2/3	0	0	1/3	0
	C	0	1	1/3	0	0
	G	1/3	0	0	2/3	0
	T	0	0	0	0	1
	-	0	0	2/3	0	0
	0	-3	-6	-9	-12	-15
A	-3	↖ 5/3	← -4/3	← -5/3	← -14/3	← -23/3
C	-6	↑ -4/3	↖ 11/3	← 10/3	← 1/3	← -8/3
G	-9	↑ -13/3	↑ 2/3	↑ 3	↖ 5	← 2

Figure 2-8. Application of dynamic programming to profile alignment. T (="ACG") is the column sequence, and the "row sequence" is a profile.

The second step is to perform a special version of the dynamic programming to generate the score matrix and backtrack matrix. With the length of T being 3 and the length of S being 5, there are only 15 cells to fill in. Because one needs to compute three values (DIAG, UP and LEFT) for each cell, the total number of values to compute is 45. Yet even for such a small problem, manual computation is quite difficult and error-prone.

The score matrix and the backtrack matrix (Figure 2-8) were obtained with a special scoring scheme. There are two kinds of matches, a match involving two identical nucleotides, or a match involving two gap symbols. The match score for the former and latter are designated M_{Nuc} and M_{Gap}, respectively, with corresponding values set to 2 and 1, respectively, in the example. There are also two kinds of mismatches, one involving a transitional difference and the other a transversional difference. They are designated as MM_s and MM_v, respectively, with corresponding values set to 1 and -1, respectively, in this example. In order not to make things too complicated, we use constant gap penalty with G = -3.

We now illustrate how the score matrix and backtrack matrix (Figure 2-8) are computed. For the first cell, the UP and LEFT values are simple:

UP = -3 + G = -3 – 3 = -6
LEFT = -3 + G = -3 -3 = -6

For the DIAG value, we should keep in mind that the nucleotide A in the column sequence has a probability of 2/3 of an A-A match and a probability of 1/3 of a A-G transition. This leads to

DIAG = 0 + 2/3 × M_{Nuc} + 1/3 × MM_s =5/3

Because DIAG is the maximum of the three, it is used to fill the first cell, together with the associated upleft arrow (Figure 2-8). The second cell (to the right the first cell) is simple because the profile at the second site contains C only. So the computation is the same as in regular pairwise alignment:

DIAG = -3-1 =-4
UP = -6 -3 = -9
LEFT = 5/3 - 3 = -4/3

Because LEFT is the largest of the three, the cell is filled with -4/3 together with a left-pointing arrow.

The cell likely to cause some confusion is the third, i.e., the one with a value of -5/3 and a left-pointing arrow. Note that an upleft arrow in that cell implies that A will pair with C with a probability of 1/3, penalized by MM_v = -1, and pair with "-" with a probability of 2/3, penalized by G = -3. Therefore,

DIAG = -6 - 2/3×3 - 1/3×1 = -25/3

The calculation of UP is simply UP = -9 -3 = -12. A left-pointing arrow, however, implies a gap in the column sequence, so we have a gap with a probability of 2/3 of facing a gap in the row profile, with M_{gap} = 1, and a probability of 1/3 of facing a C, with G = -3. Therefore,

LEFT = -4/3 + 2/3×1 - 1/3×3 = -5/3.

Because LEFT is the largest of the three, the cell is filled with -5/3 and a left-pointing arrow. The rest of the cells are relatively straightforward. The alignment can again be obtained by tracing the backtrack matrix (Figure 2-8):

```
AC-GT
AC-GT
GCCAT
AC-G-
```

The profile alignment outlined above represents an extension of the pairwise alignment, with the row sequence replaced by a profile. One can also replace the column sequence by a profile to align two profiles instead of two sequences. This approach is used in Clustal for multiple sequence alignment.

One might argue that the profile alignment has a serious problem as follows. T may be phylogenetically more closely related to some sequences than others in S. However, the profile alignment approach does not take this into consideration. This critique is justified. Unfortunately, alternative approaches by combining both phylogenetic reconstruction and multiple sequence alignment (Hein, 1990, 1994; Sankoff *et al.*, 1973) are generally too computationally intensive to be practical. However, recent advances in Gibbs sampler has paved alternative ways for pairwise sequence alignment (Zhu *et al.*, 1998) and multiple sequence alignment conditional on a phylogenetic tree (Holmes and Bruno, 2001; Jensen and Hein, 2005).

3.2 Multiple alignment with a guide tree

The main difficulty in aligning multiple sequences by dynamic programming is the rapidly increased need for memory and computational power. While aligning three sequences by dynamic programming has been implemented (Huang, 1993), it is not practical to align more than three sequences. For this reason, only heuristic approaches that reduce the multiple alignment problem to pairwise and profile alignment problems have been widely used for multiple sequence alignment. The most well known representative of this approach is the Clustal family of programs (Higgins and Sharp, 1988; Thompson *et al.*, 1994).

Multiple alignment in Clustal consists of three steps. The first is to perform all pairwise alignments by dynamic programming. With N sequences, there are N(N-1)/2 pairwise alignments, leading to a triangular matrix of alignment scores. The second step is to construct a guide tree by using the alignment score matrix as a sequence similarity matrix in conjunction with a clustering algorithm. Alternatively, one can convert the similarity matrix into a distance matrix and then use either UPGMA or neighbor-joining method (Saitou and Nei, 1987) to build guide tree such as the one shown in Figure 2-9. Clustal uses this latter approach. The third and final step is to traverse the node to align sequences by pairwise alignment and profile alignment (the pairwise alignments in the first step, typically an approximate one, are not reused here).

The multiple alignment starts from the most similar sequences. So we move to internal node 11 and align Seq2 and Seq6. We then move to internal node 10 and align Seq5 against a sequence profile representing aligned Seq2 and Seq6 using the method outlined in Figure 2-8. A profile is then created to represent the three aligned sequences (Seq2, Seq6 and Seq5). Moving to internal node 9, we found one child node (internal node 9) has two child nodes with two unaligned sequences (Seq3 and Seq4) which are then first aligned by using the dynamic programming method. A profile is then created to represent the aligned Seq3 and Seq4. This profile is then aligned against the profile presenting the aligned Seq2, Seq6 and Seq5. The process continues until all sequences are aligned.

It is easy to see why we should start with the most similar sequences because any alignment error will be propagated in subsequent alignment. Obviously, a wrong guide tree will bias the subsequent alignment which in turn will bias subsequent phylogenetic reconstruction based on the alignment. Unfortunately, a guide tree built from alignment scores is typically a very poor tree. For this reason, it is better to input a well established tree, whenever available, as a guide tree for multiple sequence alignment.

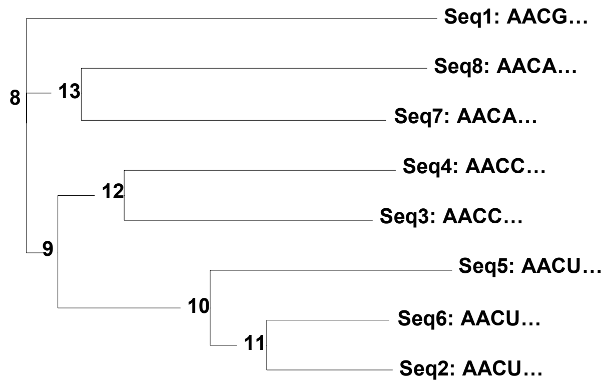


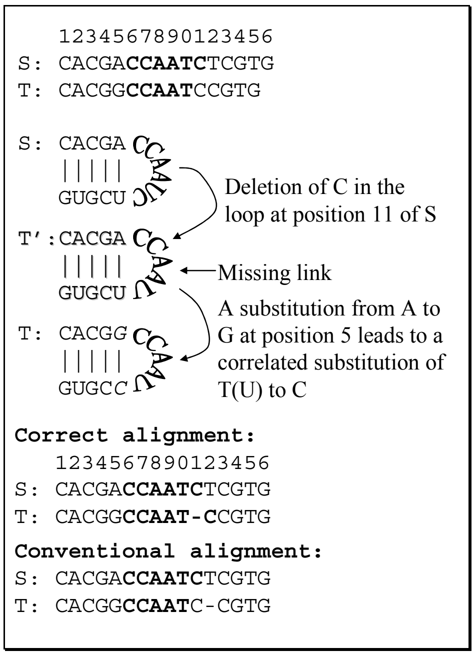
Figure 2-9. An example of a guide tree for multiple sequence alignment of eight sequences, called leaves. The internal nodes are numbered from 8 to 13 (with terminal nodes, or leaves, numbered from 0 to 7).

An alternative method of multiple sequence alignment is to use a star tree instead of fully resolved bifurcating tree as a guide tree. One starts with pairwise alignment to obtain a matrix of alignment scores in the same way as in ClustalW. This matrix is then converted to a distance matrix. We first align two sequences with the smallest distance to build the first sequence profile. The next sequence with the average distance closest to the sequence profile is the aligned to the sequence profile using the algorithm illustrated in Figure 2-8. A new sequence profile is obtained from these three sequences and the next sequence with the smallest average distance to those in the profile is aligned against the profile using the algorithm illustrated in Figure 2-8. This process continues until all sequences have been aligned.

This method is simpler than the method in ClustalW because one does not need to align two sequence profiles, i.e., in every step, the alignment is done between a sequence and a profile. The “Quick multiple alignment” function in my program DAMBE (Xia, 2001; Xia and Xie, 2001b) represents a crude implementation of this method of multiple alignment.

4. SEQUENCE ALIGNMENT WITH SECONDARY STRUCTURE

Figure 2-10 shows two sequences, S and T, being a fragment of a fictitious rRNA gene from two related species. The fragment forms a stem-loop structure. For simplicity, suppose that T was derived from S through the intermediate T’ in two steps. First, the C at position 11 was deleted. Second, a substitution from A to G at position 5 leads to a correlated substitution from T(U) to C to maintain the stem of length 5. The resulting sequence is T.



```
   1234567890123456
S: CACGACCAATCTCGTG
T: CACGGCCAATCCGTG

Correct alignment:
   1234567890123456
S: CACGACCAATCTCGTG
T: CACGGCCAAT-CCGTG
Conventional alignment:
S: CACGACCAATCTCGTG
T: CACGGCCAATC-CGTG
```

S: CACGA
GUGCU
Deletion of C in the loop at position 11 of S
T':CACGA
GUGCU
Missing link
A substitution from A to G at position 5 leads to a correlated substitution of T(U) to C
T: CACGG
GUGCC

Figure 2-10. Illustration that the correct alignment may differ from the optimal alignment. Note that T becomes U in the secondary structure.

If we constrain the alignment with the secondary structure information, i.e., the first five and the last five nucleotides of S are respectively homologous to the first five and the last five nucleotides of T, then the resulting alignment (designated as the correct alignment in Figure 2-10) correctly identifies the gap at position 11. However, any currently used alignment program based on linear sequence information, with any sensible scoring scheme, would recover the 'conventional alignment' (Figure 2-10) that identified the gap at position 12. The two C's at position 11 in the 'conventional alignment' are nicely aligned but are not homologous given the evolutionary steps given above in generating T. It is easy to see that the conventional alignment will have a greater alignment score than the correct alignment and consequently is more "optimal". Thus, optimal alignment (the alignment with the largest alignment score) may not necessarily be the correct alignment.

Aligning rRNA genes with the constraint of secondary structure has now been frequently used in practical research in molecular evolution and phylogenetics (Hickson *et al.*, 2000; Kjer, 1995; Notredame *et al.*, 1997; Xia, 2000; Xia *et al.*, 2003a). However, one cannot always assume that rRNA secondary structure is stable over time. It is now well established that variation in rRNA secondary structure is strongly affected by the optimal growth temperature in prokaryotes (Galtier and Lobry, 1997; Hurst and Merchant, 2001; Nakashima *et al.*, 2003; Wang and Hickey, 2002; Wang *et al.*, 2006).

5. ALIGN NUCLEOTIDE SEQUENCES AGAINST AMINO ACID SEQUENCES

During the evolution of protein-coding genes, an entire codon or multiple codons may be deleted or inserted, but it is much rarer to see an insertion or deletion (often abbreviated as indel) of one or two nucleotides because such indel events lead to frameshifting mutations that almost always disrupt the original protein function and are strongly selected against. However, alignment of protein-coding nucleotide sequences often produce indels of one or two bases as alignment artifacts. The correctly aligned sequences should have complete codons, not one or two nucleotides, inserted or deleted.

One way to avoid the above alignment problem is to align the protein-coding nucleotide sequences against amino acid sequences, which was implemented in software DAMBE (Xia, 2001; Xia and Xie, 2001b). The approach obviously requires amino acid sequences which can be obtained in two ways. First, if you have nucleotide sequences of good quality, then you can translate the sequences into amino acid sequences, which can be done automatically in DAMBE which implements all known genetic codes for translating protein-coding sequences from diverse organisms. Second, if you are working on nucleotide sequences deposited in GenBank, then typically you will find the corresponding translated amino acid sequences.

The alignment of protein-coding nucleotide sequences is typically done in three steps. First, the nucleotide sequences are translated into amino acid sequences. These amino acid sequences are then aligned, and the nucleotide sequences are then aligned against the aligned amino acid sequences.

Here is a simple illustration. Suppose we are to align the following two protein-coding sequences designated S1 and S2, respectively:

```
S1 ATG CCG GGA TAA
S2 ATG CCC GGG ATT TAA
```

Step 1: Translate the sequences into amino acid sequences (one-letter notation) to get:

```
S1 MPG*
S2 MPGI*
```

Step 2: Align the amino acid sequences:

```
S1 MPG-*
S2 MPGI*
```

This alignment implies the deletion of an amino acid (and its associated codon) just before the termination codon.

Step 3. Align the protein-coding nucleotide sequences against aligned amino acid sequences. This is done by essentially mapping the codon sequences to the aligned amino acid sequences. Keep in mind that a gap in the aligned amino acid sequences correspond to a triplet gap in nucleotide sequences:

```
S1 ATG CCG GGA --- TAA
S2 ATG CCC GGG ATT TAA
   *** **  *** *   ***
```

This alignment, designated as Alignment 1, has 10 matches, 2 mismatches, and 1 gap of length 3. Recall that the main objective of sequence alignment is to identify homologous sites and it is important to note that different alignments may lead to different interpretations of sequence homology. With the alignment above, sites 6 of the two sequences (G in S1 and C in S2) are interpreted as a homologous site, so is site 9 (A in S1 and G in S2). These interpretations are not established facts. They are only inferences of what might have happened.

Depending on the scoring scheme, a nucleotide-based sequence alignment, i.e., without using aligned amino acid sequences as a mapping reference, may well generate the following alignment designated Alignment 2, with 12 matches, 0 mismatch and two gaps of lengths 1 and 2, respectively:

```
S1 ATG CC- GGG A-- TAA
S2 ATG CCC GGG ATT TAA
   *** **  *** *   ***
```

Note three different interpretations of the homologous sites between Alignment 1 and Alignment 2. First, the nucleotide G at site 6 of S1 is now interpreted to be homologous to the nucleotide G at site 7 of S2. Second, the nucleotide A at site 9 of S1 is now interpreted as homologous to the nucleotide A at site 10 of S2. Which of the two alignments makes more sense? If Alignment 1 is correct but we used a nucleotide-based alignment method and end up with Alignment 2, then the estimation of the genetic distance between the two sequences will be biased. The genetic distance measures the evolutionary dissimilarity between two sequences, often estimated by ignoring the indel sites. It is often used as an index of sequence divergence time in molecular phylogenetics, when calibrated by fossils with known divergence time. In this particular case, if we perform site-wise deletion of indels, then S1 and S2 would appear more similar to each other in Alignment 2 than in Alignment 1. Biased estimation of the genetic distance often results in failure in molecular phylogenetic reconstruction.

Given that a protein-coding gene is unlikely to remain functional after two consecutive indel mutations as in Alignment 2, we may argue that Alignment 1 based on the alignment of amino acid sequences is better than Alignment 2. However, there are also cases where a nucleotide-based alignment is better. Now consider the following two protein-coding sequences:

```
     3   6   9  12  15
S1 ATG CCC GTA TAA
S2 ATG CCC GTG TTA TAA
```

Of the following three alignments, designated as Alignment 1, Alignment 2 and Alignment 3, all involving one indel of length 3, which one makes more sense to you?

```
Alignment 1:
S1 ATG CCC GTA --- TAA
S2 ATG CCC GTG TTA TAA
```

```
   *** *** **       ***

Alignment 2:
S1 ATG CCC GT- --A TAA
S2 ATG CCC GTG TTA TAA
   *** *** **     * ***

Alignment 3
S1 ATG CCC G- --TA TAA
S2 ATG CCC GTG TTA TAA
   *** *** *     ** ***
```

Alignment 1 is the outcome of aligning nucleotide sequences against aligned amino acid sequences, and the other two alignments are from nucleotide-based alignments. The three alignments represent three alternative hypotheses, all involving a gap of length 3, but differ in the position of the gap. Alignment 1 has only 11 matches, whereas the other two alignments each have 12 matches. In this case, the two hypotheses represented by alignments 2 and 3 is more likely true than alignment 1. In short, aligning protein-coding sequences against aligned amino acid sequences is not necessarily better than aligning the nucleotide sequences directly unless the latter produces frameshifting deletions or insertions.

6. POSTSCRIPT

A student has once told me that it is humiliating to have human gene sequences aligned against those of chimpanzees, monkeys or even snakes and turtles. I suspect that people of the past would also find it humiliating to have our earth displaced from the center of the universe and ranked among the other planets. Indeed it would have been much nicer, at least esthetically more charming and spiritually more enlightening, to have human genes all quite different from genes of all other living creatures. It would also have been nicer to have our earth centered in the universe with all stars and galaxies orbiting around us. That would instill into our mind a certain confidence that a supernatural custodian is looking after our wellbeing. We would have been more coherent when preaching to our children.

But nature, as beautiful as she is, does not always seem to be our maid working according to our dictation or wistful thinking. We are part of nature's creation and have been shaped by the same two sculptors of biodiversity called mutation and selection as all the other creatures. From this perspective we can better appreciate the truth that not only all humans are created equal, but also are all other creatures populating the earth. Sequence alignment is a powerful tool to help us position ourselves properly in the nature of things.

Chapter 3

CONTIG ASSEMBLY

1. INTRODUCTION

A contig is defined as a contiguous sequence assembled from a set of sequence fragments, typically by string matching and local sequence alignment. Contig assembly refers to the process of assembling many sequence fragments into one long genomic sequence or a few long contigs (Figure 3-1). Although the first automatic sequencer was developed by Leroy Hood in 1986 and the Human Genome Project was in full swing in 1990, all automatic sequencers still suffer from the problem of limited read length of ~700 bases. Consequently, any genome-sequencing project will always involve the sequencing of a large number of sequence fragments. These fragments need to be assembled to contigs and finally to a complete genomic sequence.

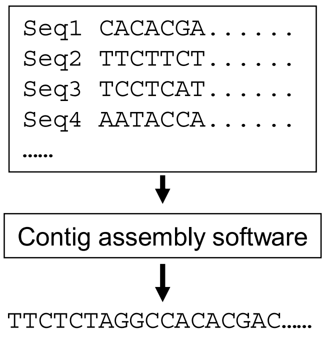


Figure 3-1. Contig assembly

A contig is a reconstructed nucleotide sequence from more than one sequence fragment. The fragments are assembled by identifying overlapping sequence fragments based on local string matching and alignment methods that identify the overlapping ends of the sequence fragments and statistical methods that evaluate the significance of the matched sequence ends. A computer program for contig assembly will take as input a set of sequence fragments, and output one or more reconstructed contigs. It is important to recognize the fact at the very beginning that a reconstructed contig or genomic sequence may not be the same as the real sequence.

Contig assembly is really the first step in the pipeline of genomic analysis and, in this sense, should have been covered in the first chapter of a book on bioinformatics. However, contig assembly requires the understanding of algorithms in string matching and sequence alignment. For this reason, the subject is better covered after the reader has gained familiarity with string matching covered in Chapter 1 and sequence alignment covered in Chapter 2.

Two types of contig assembly programs are in current use. The first is the Phred/Phrap/Consed combo (Gordon *et al.*, 1998) used in hierarchical shotgun sequencing (e.g., the government-sponsored Human Genome Project) together with post-assemblers such as the GigAssembler (Kent and Haussler, 2001) to merge the contigs into a genome. This divide-and-conquer approach cuts genomes into mapped mega pieces which are further cut into mapped smaller pieces. So contig assembly involves assembling the sequence fragments derived from each small genomic segment. The second is the Celera (Myers *et al.*, 2000; Weber and Myers, 1997) and PCAP (Huang *et al.*, 2003) assemblers for whole-genome shotgun sequencing. In this case, contig assembly has to deal with all fragments derived from the entire genome.

The contig assembly algorithm used in these programs involves three steps. The first is to perform pairwise matching of sequence ends (including the complimentary sequences) to identify “mate pairs” with the 3’-end of one mate overlapping the 5’-end of the other. The confidence of the overlap can be assessed by statistical methods detailed in the first chapter that has also been used to assess the significance of an exact string match in BLAST (Altschul *et al.*, 1990; Altschul *et al.*, 1997) and FASTA (Pearson, 1998). The confidence is enhanced by searching the putative mate pair against EST databases to see if both overlap the same segment of an EST. The Goldenpath human genome assembly is produced by first building an initial set of sequence contigs and then using paired plasmid ends, ESTs, BAC end pairs, etc., to order and orient these individual contigs into larger assemblies (Kent and Haussler, 2001).

The second step is the actual assembly. If a sequence fragment is involved in only one or two mate pairs, then assembling these mate pairs is unambiguous. Contig assembly of sequence fragments from prokaryotic genome sequencing projects is generally unambiguous because of the rarity of sequence repeats in prokaryotic genomes. Sequence repeats are frequent in eukaryotic genomes, and a fragment can be involved in many more than two mate pairs. For example, if fragment i is really flanked by fragments j and k in a eukaryotic genome, but fragment j has N repeats in the genome, say fragments j_1, j_2, ..., j_N, then fragment i would be involved in at least N mate pairs. Now we will have a hard time deciding which of the putative mate pairs represents true neighborhood relationship.

Both greedy and non-greedy algorithms are used to treat the resulting putative pairs. The greedy one will assemble the two mates into one. If the assembly of the two is wrong, the error will be propagated to subsequent longer contigs. A non-greedy algorithm will generate a directed graph based on pairwise matching. We will have more details on the non-greedy algorithm latter in the chapter.

The third step is to resolve conflicts. As we will see latter, sequence repeats can lead to multiple alternative assembled contigs. It is important to have a criterion to assess alternative assemblies.

Contig assembly involves a great deal of mathematics and computational algorithms. We will first provide a skeletal outline of the contig assembly and then explain each step with more details.

2. SKELETAL OUTPUT OF CONTIG ASSEMBLY

Given a set of N sequence fragments, we first obtain N complementary sequence fragments. The necessity of this step is shown in Figure 3-2 where two sequence fragments labeled 2 and 6 were from the top strand and four sequence fragments were from the bottom strand. The 3’-end sequence of fragment 3 is the same as the 5’-end of the complementary sequence of fragment 2, but not the same with any part of fragment 2 itself. Similarly, the 5-end of fragment 5 is the same as the 3’-end of the complementary sequence of fragment 6. Any practical contig assembly will involve many fragments, but we will use the fragments depicted in Figure 3-2 for subsequently illustrations.

In contig assembly literature, the two DNA strands are often referred to as the plus strand and the minus strand. When we have N sequence fragments and N complementary sequence fragments, contig assembly ideally should generate a plus strand and a minus strand that will be complementary to each other. This can be used for validating the contig assembly algorithm.

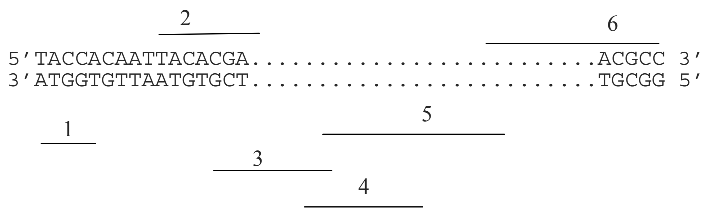

2
6
5'TACCACAATTACACGA.........................ACGCC 3'
3'ATGGTGTTAATGTGCT.........................TGCGG 5'
5
1
3
4

Figure 3-2. Six sequence fragments from the double-stranded genomic sequence. Fragments 2 and 6 are collinear with the top DNA strand whereas fragments 1, 3, 4 and 5 are collinear with the bottom DNA strand.

Contig assembly starts by doing pairwise comparisons between fragments i and j. Each pairwise comparison consists of (1) matching the 3'-end of fragment i with 5'-end of fragment j and (2) matching the 5'-end of fragment i with 3'-end of fragment j. Each match is evaluated for its statistical significance. The result of these N*(N-1)/2 pairwise comparisons is summarized in a matrix of matching scores (Figure 3-3). Note that a practical contig assembly in a shotgun sequencing project will typically involve hundreds of thousands of sequence fragments and consequently a matrix with many millions of elements. However, only good scores are kept for identifying mate pairs. So the actual number of scores that need to be kept in computer memory is much smaller.

	2	3	4	5	6
1	P	P	P	P	P
2		G	P	P	P
3			G	G	P
4				G	P
5					G

Figure 3-3. Matrix of matching scores for assembling the six fragments in Figure 3-2. Symbols P and G stand for poor and good matching scores, respectively. Only good matching scores need to be kept.

In actual computation, the simplest generic data structure for individual sequence fragment is as follows:

```
DataStructure dsFragment
  dsFragment fragmentWhose3EndMatchingMy5End()
  int goodMatchScore1()
  int arrayDimension1
  dsFragment fragmentWhose5EndMatchingMy3End()
  int goodMatchScore2()
  int arrayDimension2
End DataStructure
```

It is worth noting the difference between two kinds of string matches. The first is the overlapping match (Figure 3-4) used in contig assembly, and the second is the non-overlapping match (Figure 3-4) that is marked and discarded. Relevant statistical methods for evaluating the quality of an overlapping match have already been presented in the first chapter.

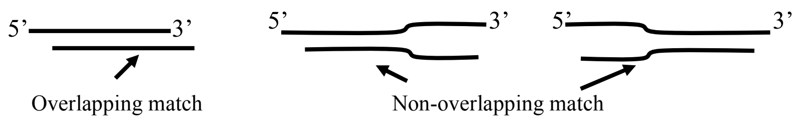


Figure 3-4. Contig assembly uses only overlapping matches.

From the pairwise comparisons of the six sequence fragments and their complements, we obtain the matrix in Figure 3-3 and an array of dsFragments from which one can construct a graph to obtain the contigs (Figure 3-5) by following the directed path. Note that there are two paths. The first is made of fragments 2, 3, 5 and 6 in the order of 2→3→5→6 and the second is made of fragments 2, 3, 4, 5, and 6 in the order of 2→3→4→5→6. Both lead to the same assembled contig. Fragment 1 has no arrow to link it to any other fragments and will not be assembled with any other fragment.

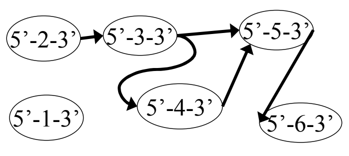
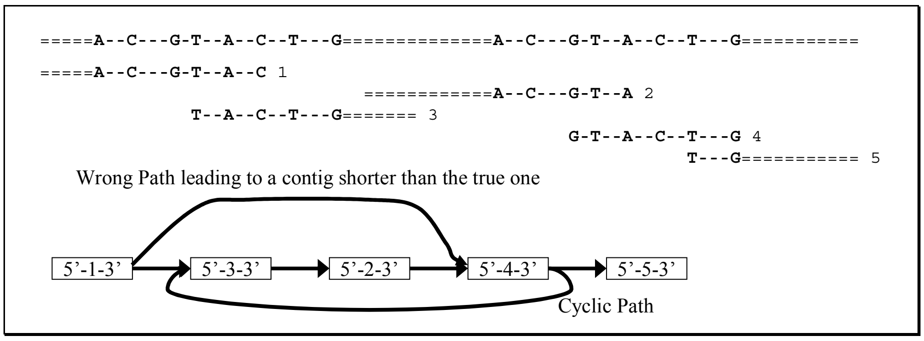


Figure 3-5. Directed graph constructed from the matrix of good matching scores in Figure 3-3. Only good (statistically significant) scores contribute to the graph construction. An arrow directing from fragment i to fragment j means that the 3-end of fragment i overlaps the 5'-end of fragment j with a good score, i.e., each arrow links a mate pair.

There are two major difficulties with the non-greedy algorithm (Figure 3-5). The first is the large number of possible pathways to construct the contig. Take a sequencing project with 10× coverage for example ("10× coverage" means that the total length of all sequenced fragments is 10 times as long as the estimated genomic length). Each node in the directed graph will on average have 10 connections, with some nodes having many more. As the number of possible paths increases rapidly with increasing number of sequence fragments, it becomes difficult to decide which path to follow. We all know the difficulty when encountering two roads that diverged in the yellow wood, but we cannot afford the luxury of idly gazing down both as far as we could. We have to decide which paths are likely the right ones and let the computer help us travel down all of them and finally check to see if they are all the same. Fortunately, modern computers can go very fast, especially when your instruction is clear.

The second problem with the non-greedy algorithm (Figure 3-5) results from sequence repeats leading to (1) spurious connections and conflicting paths and (2) cyclic paths in the directed graph that trap the program that follows the paths to obtain contigs (Figure 3-6). The cyclic path is produced because the 3'-end of fragment 4 (i.e., T--A--C--T---G) is the same as the 5'-end of fragment 3. The path that would lead to a shortened contig is caused by the 3'-end of fragment 1 (G-T--A—C) being the same as the 5'-end of fragment 4.


```
=====A--C---G-T--A--C--T---G==============A--C---G-T--A--C--T---G===========

=====A--C---G-T--A--C 1
                              ============A--C---G-T--A 2
              T--A--C--T---G======= 3
                                                 G-T--A--C--T---G 4
                                                            T---G=========== 5
```

Figure 3-6. The problem of sequence repeats in contig assembly. The top is the true genomic sequence with one segment repeated (shown in bold). The five fragments, with numeric labels to the right, were obtained from the genomic sequence.

The conventional approach to solve conflicting assemblies is to use the Bellman-Ford algorithm to find the shortest path (Kent and Haussler, 2001; Thayer *et al.*, 1999). However, the shortest path does not imply the correct assembly. As illustrated in Figure 3-6, the shortest path will result in omission of repeated sequences. The human and mouse genomes from Celera, assembled from the whole-genome shotgun sequencing that is relatively vulnerable to sequence repeats, have many fewer sequence repeats (which most likely represent artifacts from contig assembly) than the one from Human Genome Project (Istrail *et al.*, 2004). There are also substantial differences between different human genome assemblies (e.g., between NCBI and University of California at Santa Cruz, or UCSC) from public effort (Rouchka *et al.*, 2002). While these different public assemblies have now been coordinated to provide a unique human genome assembly, it is not clear how discrepancies are resolved. A discrepancy between the two may mean both being wrong or one being wrong and one right. Forcing a consensus does not always mean that the result will be right.

Once when I get to this point in the classroom, a student suddenly asked why the two public assemblies could not both be right. My answer was that both assemblies were from the same source of fragments and therefore could not both be right when there were substantial differences. Whenever there was a substantial difference, either assembly A is wrong or assembly B is wrong or both are wrong. The student could not understand the logic and I, at my wit's end, suddenly recalled what Voltaire said in his Philosophical Dictionary that, given the evil we observe in this world, "Either God wishes to expunge the evil from this world and cannot; or He can and does not wish to; or he neither wishes to nor can." Equipped with this sudden rush of insight, I explained that "both assemblies being correct given the differences" was logically similar to "God wishes and can, given the observed evil", both being logically impossible. The student wanted me to show the connection between the two statements. Taking it literally, I draw a circle containing "both assemblies being correct given the differences" in the left of the paper and another circle containing "God wishes and can given the observed evil" in the right of the paper and draw an arrow from the left circle to the right circle. Unfortunately, the student claimed that this was not the connection she wanted. In the end, I was left more confused than the student. (Voltaire probably did not read Isaiah 45:7, King James Version, where one finds God stating "I form the light and create darkness, I make peace and I create evil, I am the LORD who does all of these". All religions share the notion of the good and the evil, especially oriental ones which are rich in dialectical thoughts. According to dialectics, the good cannot be defined or known without the evil. In contrast to the first law of logic, i.e., the law of identify which states that A is A and cannot be anything else, the first law of dialectics is the law of the negation of negation which states that A can be A only if it is not a non-A. In other words, A cannot be A unless the presence of a non-A is a *condicio sine qua non*. Therefore, God cannot create the good without creating the evil at the same time, or create light without creating darkness at the same time. We cannot see anything when confined in total darkness or when staring into pure light. A balance of light and darkness is necessary for us to see. For the same reason Solomon had wisely stated in Ecclesiastes 7:16-17 that "Be not overly righteous, …… Be not overly wicked". In this sense, the Bible is quite dialectical. From this point of view we see that Voltaire was logical but not philosophical.)

Another problem with contig assembly, other than those getting both my students and me confused, is that the ending sequences are often of poor quality, making it difficult to have exact matches. However, modern automatic DNA sequencers can perform quality analysis and automatically chop off sequence ends that are poor.

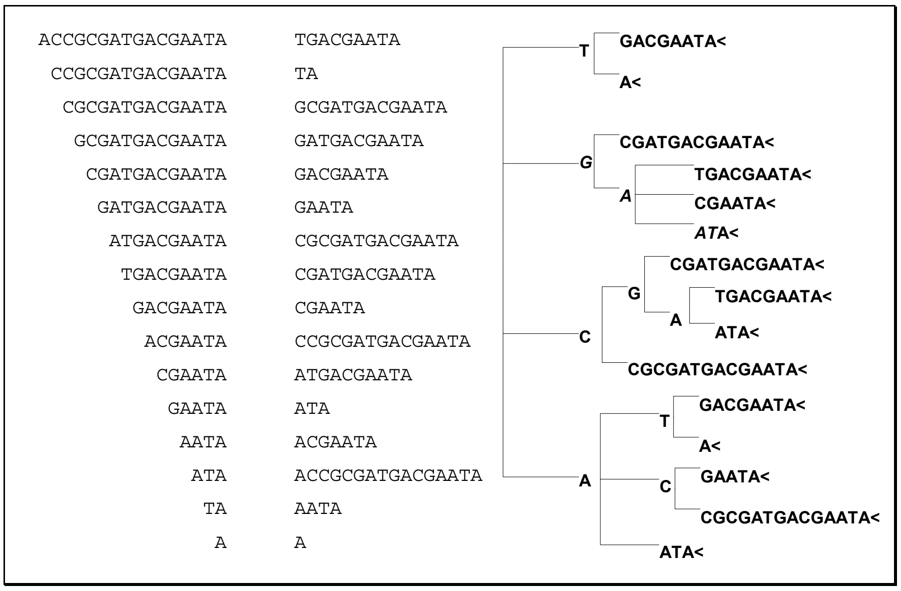
3. STRING MATCHING OF TWO SEQUENCE ENDS

Contig assembly needs to do N×(N-1)/2 pairwise comparisons of the ending sequences to identify overlapping matches. With N in the order of 10,000-100,0000 or more, the computational burden is huge and efficient string matching methods are needed. String matching algorithms most familiar to biologists are BLAST (Altschul *et al.*, 1990; Altschul *et al.*, 1997), FASTA (Lipman and Pearson, 1985), and the global alignment by dynamic programming (Needleman and Wunsch, 1970) popularized by Clustal (Higgins and Sharp, 1988).

String matching algorithms are classified into exact and inexact matching (Gusfield, 1997). Commonly used string matching algorithms are hash tables which have been covered in the first chapter, suffix trees (McCreight, 1976; Ukkonen, 1995; Weiner, 1973) which is a data-structure that facilitates rapid finding of an exact match of a substring in a string, and pairwise alignment by dynamic programming (Needleman and Wunsch, 1970). Hash tables and suffix trees are representatives of the exact string matching algorithm, whereas sequence alignment by dynamic programming is a representative of the inexact string matching algorithms. We have covered hash tables and sequence alignment by dynamic programming in previous chapters. Here we briefly mention the suffix tree.

If S = $t_1t_2...t_i...t_n$, then $S_i = t_it_{i+1}...t_n$ is the suffix of S that starts at position i. The suffixes of a string (S = ACCGCGATGACGAATA) are shown in the left column in Figure 3-7. S is a suffix of itself starting at position 1. The middle column (Figure 3-7) contains the sorted suffixes, and the right panel is a suffix tree. A related concept is a trie (from re**trie**ve) in which each nucleotide is a node on the tree, with branches linking its neighboring nodes (e.g., the top leaf 'GACGAATA<' would have been written as 'G-A-C-G-A-A-T-A<').

A suffix greatly decreases the time needed to search one string of length M against another string of length N. Optimized searching algorithms, such as the Boyer-Moore algorithms (Gusfield, 1997), can finish the search in no more than M+N comparisons. The suffix tree need only M character comparisons.



ACCGCGATGACGAATA	TGACGAATA
CCGCGATGACGAATA	TA
CGCGATGACGAATA	GCGATGACGAATA
GCGATGACGAATA	GATGACGAATA
CGATGACGAATA	GACGAATA
GATGACGAATA	GAATA
ATGACGAATA	CGCGATGACGAATA
TGACGAATA	CGATGACGAATA
GACGAATA	CGAATA
ACGAATA	CCGCGATGACGAATA
CGAATA	ATGACGAATA
GAATA	ATA
AATA	ACGAATA
ATA	ACCGCGATGACGAATA
TA	AATA
A	A

Figure 3-7. Suffixes of S = ACCGCGATGACGAATA (left column), alphabetically sorted suffixes (middle) and a suffix tree (right). The trailing '<' indicates the ending character node.

Suppose now we wish to find whether the 5'-end of sequence T (= GAATACGACTGACGATGGA) matches the 3'-end of S. We can start with the first four nucleotides of T, i.e., GAAT. By referring to the suffix tree in Figure 3-7, we can quickly find GAAT (italicized letters in the suffix tree). Because the last 'T' in GAAT is not the ending character node (Figure 3-7), we need to extend the match until we encounter the ending character node. In this case, the extension is successful because the last character node A on the suffix tree matches the fifth character of T. So we conclude that S and T have an overlapping match of length five. If our extension fails before reaching the ending character node, then the match is not an overlapping match, i.e., it is one of the non-overlapping matches depicted in Figure 3-4. Such non-overlapping matches are not used to generate mate pairs.

The trie structure and suffix trees are used frequently in storing large dictionaries in spell-checking programs, where many branches may originate from one node (e.g., 26 branches may originate from a node in the English language). The trie or suffix tree for DNA sequences has at most four branches originating from each node.

An alternative string matching method is illustrated in Figure 3-8, using the same two strings S and T. The binary coding is similar to that used in BLAST (Altschul *et al.*, 1990; Altschul *et al.*, 1997), i.e., A:00, C:01, G:10, T:11, with each nucleotide taking up two bits. This allows four nucleotides to be packed into a single byte (8 bits). To know whether the 3'-end of S matches the 5'-end of T (or any other sequence), we obtain a list of suffix numbers of S starting at the last byte of S (i.e., 00001100) and shifting leftwards with the step length of two bits (i.e., one nucleotide). This is illustrated in Figure 3-8. Note that the binary number of 00001100 is equal to the decimal number of 12:

$$\begin{aligned}00001100 &= 0\bullet 2^7 + 0\bullet 2^6 + 0\bullet 2^5 + 0\bullet 2^4 + 1\bullet 2^3 + 1\bullet 2^2 + 0\bullet 2^1 + 0\bullet 2^0 \\ &= 0+0+0+0+8+4+0+0=12\end{aligned} \tag{3.1}$$

```
S: ACCGCGATGACGAATA
T: GAATACGACTGACGATGGA
Recoding A:00, C:01, G:10, T:11
S: 00010110011000111000011000001100
T: 10000011000110000111100001100011101000
From 3'-end of S:
    00001100        12
  1000001100        524
011000001100        1548
From 5'-end of T:
10000011            131
1000001100          524
100000110001        2097
Conclusion: The last 5 nucleotides of S
matches the first 5 nucleotides of T because
the shared number of 524.
```

Figure 3-8. A simple illustration of an exact string matching method.

Similarly, we obtain a list of prefix numbers of T starting from the first byte of T and shifting rightward with the step length of two bits. A prefix number in T that matches a suffix number in S implies an overlapping match. In our case, the number 524 is shared between the prefix numbers of T and the suffix numbers of S, leading to the conclusion of an overlapping match of 5 nucleotides (Figure 3-8). Because the suffix and prefix numbers are ordered from small to large, a search for a matching number can be done quickly. My program DAMBE (Xia, 2001; Xia and Xie, 2001b) has a contig assembly function, partially based on Huang's (1992), includes a crude implementation of these string matching methods.

When an exact overlapping match is identified by methods in the previous section, it is important to know whether the match is statistically significant or may happen by random occurrence. So we need a filter to eliminate poor matches and establish good matches. This is covered in the chapter on BLAST and FASTA.

4. NEW DEVELOPMENT IN CONTIG ASSEMBLY

There are still many unsolved or even unsolvable problems with contig assembly (Pevzner *et al.*, 2001). For this reason, all assembled genomes from multicellular eukaryotes have gone through a seemingly everlasting process of revision. However, a new development in sequencing and contig assembly in my laboratory might be worth mentioning.

Consider piecing together a jigsaw puzzle. We first would find all the edge pieces and then optionally sort pieces into similar colors. This divide-and-conquer approach is used often in efficient computation. What are the edge pieces in contig assembly?

We propose to use single-copy genes to dramatically reduce the computation time and error rate in contig assembly. In spite of genome duplication events during evolutionary time, many genes are single-copy and highly conserved in extant genomes. For example, the budding yeast has about 3500 single-copy genes (Goffeau and al., 1996). The human genome shares 1308 gene families with the genomes of *Caenorhabditis elegans*, *Drosophila melanogaster* and *Saccharomyces cerevisiae*, 43.1% of which are single-copy genes in these organisms and in humans (Hughes *et al.*, 2001; Lander *et al.*, 2001a; McLysaght *et al.*, 2002; Panopoulou *et al.*, 2003). The number of families shared between the human genome and the genomes of *C. elegans* and *D. melanogaster* increases to 3044, most of which are single-copy genes (Panopoulou *et al.*, 2003). Approximately 3,700 of the genes in the Arabidopsis Col-0 genome are single-copy, and are distributed throughout the genome.

We can exploit these single-copy genes (SCGs) in contig assembly in three steps. First, we extract all fragments matching each SCG and perform SCG-anchored contig assembly. Second, the genome can be restriction-digested to obtain the physical distance between SCGs (D_{ij}). Third, routine contig assembly can be done for the remaining sequences. When conflicting assemblies arise, we can choose the assembly that fits best the D_{ij} values. This criterion is obviously superior over the shortest-path criterion in the Bellman-Ford algorithm used in current contig assemblers. This contig assembly strategy, once implemented in a user-friendly manner, should dramatically increase the efficiency of contig assembly and, in particular, improve the accuracy of the assembled genome and the quality of all downstream genomic analyses.

5. POSTSCRIPT

A genome typically has many genes, and genes and gene products interact with each other in many ways in a living cell, creating many different relationships among them. Contig assembly identifies one of the relationships among genes, the neighborhood relationship.

The importance to understand the relationships among genes has been highlighted by using the Greek legend of the Delphi boat (Danchin, 2002). Delphi is the location where the son of the Greek god Zeus, Apollo, established a priesthood. The female priests could often enter into a mental state in which they were prone to murmur an encoded prophecy of the future called an oracle. For this reason the word Delphi and Oracle have been associated with prophecy of the future and have both been adopted by software companies. The female priests might also ask passersby questions, and one of the question is whether a boat made of wooden planks remains the same boat after all of its original planks have gone rotten over time and been sequentially replaced by new planks. From the owner's point of view, the boat is the same boat because it is not the individual planks that define the boat, but the relationships among planks, as well as the constraints imposed by these relationships on the size and shape of individual planks, that define the boat. While having a list of planks is an essential step in building a boat, it is only after we have identified the relationship among planks can we actually build the boat.

The relevance of the Delphi boat to genomics is that, although we often claim to be in the postgenomic era, all we really have at the moment is really just a list of genes equivalent to a pile of wooden planks. We still know very little about the relationship among the genes and gene products and consequently are still far from building a real boat (a functional genome in this case). Contig assembly does help to identify the neighborhood relationship although the neighborhood relationship identified by current genomic technology is still often wrong for most multicellular eukaryotic genomes.

If the boat we are trying to build is a living cell, then we are not even close to get a good parts list. A cell is a chemical system with numerous components interacting with each other to determine the state of the cell, i.e., whether it is to become a functional liver cell or to turn cancerous. Three classes of the interacting cellular components are considered to be particularly important, the genome, the transcriptome (i.e., the collection of

RNA molecules transcribed from the genome, often defined with reference to cell type and time) and the proteome (i.e., all proteins translated from mRNAs, also often defined with reference to cell type and time). As we will learn latter, the assembled genomic sequences have greatly accelerated the characterization and quantification of transcriptomes and proteomes.

Chapter 4

DNA REPLICATION AND VIRAL EVOLUTION

1. INTRODUCTION

We have learned in the previous chapters the fundamental tools for homology searches, sequence alignment, and contig assembly to assemble sequence fragments into a genome. If you are not a biology student, you may ask why we take all this trouble to sequence and assemble genomes. To this I have two answers. The first is that we, at this point in the book, are still at the early stage of data collection in genomics. The tremendous value of obtaining well assembled genomes will become obvious when we progress further into the book and have acquired more biological knowledge and genomic analysis such as descriptive, comparative and functional genomics. The second answer, which is more relevant to your question, is that the genomic sequences, even not annotated, can already be used to derive biological insights and to test interesting predictions derived from biological hypotheses.

This chapter, together with Chapter 8, serves three purposes. First, they will not only provide essential biological concepts, especially for non-biology majors, to pave the way for studying latter biological problems, but also present these concepts in the framework of molecular evolution shaped by the three essential biological processes: genome replication, transcription and translation. Without such biological concepts we are likely to get lost in the jungle of disconnected biological entities. Second, they illustrate biological research that can be done with molecular sequences, you will soon realize that the less we know about the genome, the fewer inferences we can make. In fact, you will soon realize that inferences built upon uncertain inferences makes you feel uncomfortable very quickly. Third, we need to have some variation after three chapters of computational algorithms. An ancient Chinese administrator (晏子 or Yan Zi) recognized the importance of variation and diversity about 2500 years ago. When asked by the king for his opinion on another official who always agreed whole-heartedly with the king, 晏子 answered that, although water is essential in cooking, adding water to water can never generates gourmet food (若以水济水，谁能食之？). So I should give you something more than just plain water.

In this chapter, we will focus on DNA replication to show that even unannotated genomes can be useful in addressing biologically interesting questions, i.e., the relationship between bacteria and their parasites, the bacteriophage (or phage for short). Just in case the reader happens to have only limited knowledge in biology, I will outline basic viral and bacterial biology and then perform a very limited genomic analysis based on one of the simplest genomic properties, the genomic GC% (often referred to as GC content), to see how mutation and selection can shape viral genomes.

I have to say here that I am hesitant about including this chapter because of dramatically different feedbacks from colleagues, with one condemning the chapter by saying that it is like an ugly tumor on an otherwise beautiful nymph. As tumors are featured far more prominently in science than nymphs, I hope that you would not mind seeing a tumor sticking out somewhere.

2. FUNDAMENTALS OF VIRUSES

Sir Peter Medawar has described the virus as “a piece of bad news wrapped up in protein” (Medawar and Medawar, 1983, p. 275), and the “bad news” comes in a variety of colors and shapes. The viral diversity serves as a bridge connecting the organic chemicals to living organisms. In one extreme of this diversity, nature presents us with some “bad news” that are never observed to have been wrapped in a protein coat, and such “bad news” are called viroids that are infectious agents of plants comprised of just a single-stranded circular RNA molecule of about 300 nucleotides (Gora-Sochacka, 2004). On the other extreme of viral diversity, nature also features some giant viruses, unfortunately called miniviruses, that blurs the distinction between viruses and the parasitic cellular organisms (La Scola *et al.*, 2003; Raoult *et al.*, 2004). The largest known minivirus has a 400-nanometer particle size that is comparable to Mycoplasma and a genome size of 1,181,404 bp that is longer than that of *Mycoplasma genitalium* whose genome size is only 580,074 bp (Raoult *et al.*, 2004).

Viroids are not considered as viruses, and mimivirus exhibits many features that distinguish it from other large DNA viruses (Raoult *et al.*, 2004). This chapter will cover viruses that lie between these two extremes.

2.1 The virion and the viral genome

A viral particle outside of a living cell is called a virion, which consists of an outer shell and an interior core. The outer shell, made of capsid proteins, protects the contents of the core and has proteins interacting with the receptor on the cell membrane of the host cell, therefore conferring the host specificity. For example, HIV-1, the cause of AIDS, binds to the chemokine receptor CCR5 found on human lymphocytes and macrophages. The interior core contains the viral genome and, in some viral species, one or more enzymes needed to start the process of reproduction within the host cell. The genome encodes proteins needed for viral reproduction but not provided by the host cell.

The smallest virus is probably the hepatitis B virus with a genome size of ~3 kb and a virion particle size of ~42 nm. The parvoviruses have smaller capsids (18-26 nm), but a larger genome (5 kb)

Just like the more complicated cellular organisms, a virus needs all three essential biological processes, i.e., genome replication, transcription and translation, to reproduce itself. It needs to orchestrate these three processes to be efficient. Take phage λ for example. If the phage would generate many copies of its genome but not a corresponding number of head and tail proteins, or if it would generate too many tail proteins but too few head proteins, then it would simply be too wasteful to be tolerated by nature. According to Charles Darwin (Darwin, 1859), our mother nature is not particularly forgiving for those straying away too far from life's imperatives of passing one's genome to the next generation.

The information for orchestrating the three essential biological processes is stored in the genome, and different viruses have different ways of storing the information. Viral genomes can be classified into four types: double-stranded DNA (dsDNA), single-stranded DNA (ssDNA), double-stranded RNA (dsRNA) and single-stranded RNA (ssRNA). The four types of viruses differ significantly in the genome size. Based on the 118 bacteriophage (which are viruses parasitizing their bacterial hosts) genomes retrieved from NCBI on September, 2003, with 79 dsDNA phage species, 27 ssDNA phage species, 4 dsRNA phage species and 8 ssRNA phage species, the average genomic sequence lengths are 39,297, 6,274, 6,658, and 3,801 for dsDNA, ssDNA, dsRNA and ssRNA, respectively.

2.2 Variation in viral genome size can be explained by variation in mutation rate

The variation in genome size is explained by different mutation rate operating on different genomes. It is necessary here to introduce the concept of fitness in evolutionary biology. Fitness of an individual is defined theoretically as the propensity of leaving offspring in future generations, and is measured operationally in two ways. The first, called absolute fitness is measured by the number of offspring brought up to breeding age. The second, called relative fitness, is typically measured as the ratio the absolute fitness of a mutant over the absolute fitness of a wild type individual. J. B. S. Haldane (1937) argued that, in equilibrium populations, the effect of deleterious mutation on average fitness depends primarily on the mutation rate and is independent of the severity of the mutations. In particular, the average fitness ($\overline{w}$) at equilibrium depends on the genomic mutation rate according to the following equation

$$\overline{w} = e^{-\mu L} \tag{4.1}$$

for haploid or asexual diploid populations (Hopf *et al.*, 1988; Kondrashov and Crow, 1988), where μ is the deleterious mutation rate per nucleotide site and L the genomic length. For competing species that co-exist for a long time, the average species fitness should all be 1, i.e.,

$$e^{-\mu L} \approx 1 \tag{4.2}$$

which means that, on average, each genome should produce an equally mutation-free genome. This implies that a species with a large μ will necessarily have a small L because a genome with a large μ and a large L would reduce the fitness ($\overline{w}$)so that the carrier of such a genome will be eliminated by natural selection. The late John Maynard-Smith has taken a slightly different modeling approach but reached the same conclusion (Maynard Smith, 1989, pp.20-24).

That μL should be approximately constant for genomes in haploid organisms and asexual diploid organisms has been well documented empirically (Table 4 in Drake *et al.*, 1998). Mutation rate μ is relatively lower in dsDNA than in ssDNA or in RNA viruses, which explains why dsDNA viral genomes can be substantially longer than other viral genomes. It is mainly because of the low mutation rate in dsDNA viruses that efficient vaccines can be developed against them to protect us from their infection.

One interesting exception to this rule is the Hepatitis B viruses which are dsDNA viruses but with a very small genome of ~3 kb. The genomic replication in Hepatitis B viruses involves a lengthy mRNA stage. In other words, the genome is copied to mRNA molecules which not only serve as templates for making proteins, but ultimately also serve as templates for making a complementary DNA strand. This lengthy mRNA stage means that the mutation rate should be high, and we can predict that its genome size should be small. Indeed, the hepatitis genome, with a length of just about 3 kb, is among the smallest viral genomes. This is a good example in which an exception to the rule actually supports the rule in a different way.

2.3 A representative virus: Phage λ

The phage λ virion, once attached to the cell wall of its host (*Escherichia coli*), injects its DNA into the bacterium (Figure 4-1). The linear viral DNA is then circularized into a supercoil (over-twisted DNA molecule). If the host cell is in good shape, then the phage will often initiate its lysogenic cycle by integrating its DNA to the circular bacterial chromosome and have its genome co-replicated with the host genome. In its integrated form, the viral genome is called a provirus (or prophage). Not all dsDNA viruses have the lysogenic cycle.

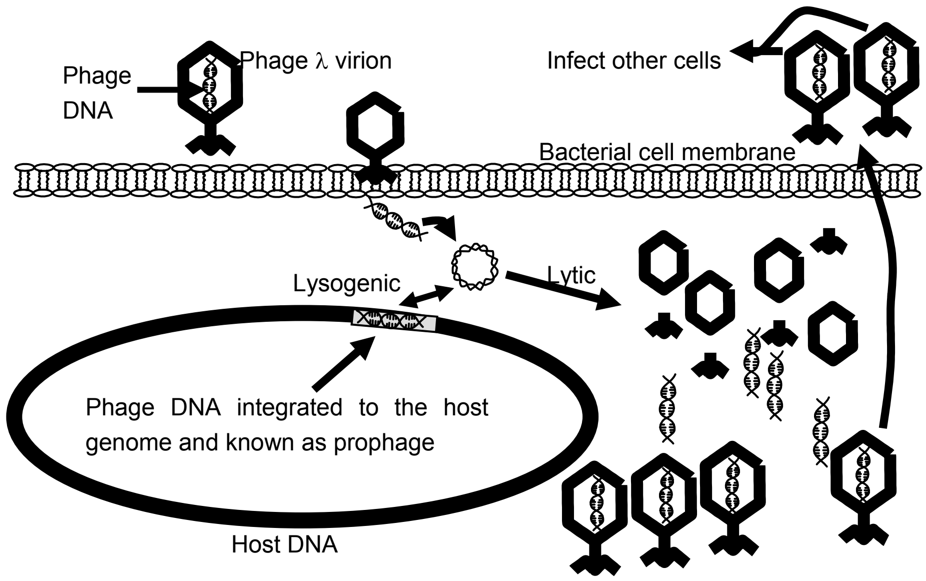


Figure 4-1. Schematic illustration of the lytic and lysogenic cycles of the phage lambda

Under certain conditions, typically when the host cell is not in a good shape to replicate quickly, e.g., when exposed to UV radiation, the phage

DNA is able to free itself out and enter the lytic cycle. Now the phage DNA is quickly replicated and transcribed, and the resulting mRNA quickly translated into phage proteins. The replicated phage DNA and phage proteins are assembled into new phage virions. Finally, the phage lysozyme is produced to lyse the host cell membrane to release the phage virions to initiate a new cycle of infection. It seems that phage λ is a pretty faithful practitioner of Sun Tzu’s military strategy. When the environment is deteriorating and when one can do little about it, then taking-off is the best strategy (三十六计，走为上计).

3. FUNDAMENTALS OF BACTERIAL SPECIES

Bacterial species used to be grouped into the gram-negative and gram-positive ones, with *Escherichia coli* being the representative of the former and *Bacillus subtilis* being the representative of the latter. These two are naturally the most studied bacterial species.

Competition is very intensive in the micro-world and most bacterial species need to be extremely efficient in order to stay in the game of survival and reproduction. *E. coli* cells replicates once every 20 minutes with unlimited nutrients. During this period it needs to replicate its genome of about 5 megabases (mb) long, transcribe millions of RNA molecules, and make millions of protein molecules. If some *E. coli* cells get lazy and leave their offspring at a slower rate, then they will soon be replaced by faster-replicating ones. This process is called negative selection in contrast to positive selection which preserves the fittest mutants.

Most bacterial genomes are circular DNAs made of A, C, G and T. Bacterial genomes differ dramatically in genomic AT%. It is likely that nucleotides A and T are abundant in bacterial species with an AT-rich genome because otherwise such bacterial species would be inefficient in replicate their DNA because few building blocks are available. Below we develop a more formal argument to show that genomic AT% of bacterial species is indicative of cellular AT availability. It is important to know something about the cellular environment inside a bacterial cell, especially from a phage perspective. It would be a fatal mistake if a phage squeezes its AT-rich genome into a bacterial host with few A and T available.

The importance of the environment reminds me of another story of Yan Zi (晏子) when he was serving as the prime minister of kingdom Qi (齐国). When he was visiting the neighboring kingdom Zhu, the king of Zhu and his aids plotted to insult him. During the banquet, two guards pushed a man in chains across the court. The king rose to ask what happened and was told that the man in chains was a native of kingdom Qi and was caught for stealing. The king then turned to 晏子 and asked if people of kingdom Qi loved stealing. 晏子 replied with a smile, "I heard that orange trees produce excellent fruits when growing in the south of the Huai River, but terrible fruits when growing in the north of the river – the different environments make them so. People become excellent citizens when living in kingdom Qi, but degrade to thieves when living in kingdom Zhu – the different environments make them so."

Some of my students are pretty good at invoking the same argument. They would tell me that they were A+ students when taught by other professors, but became B- students when taught by me – the different professors make them so.

4. GENOMIC AT% OF BACTERIAL SPECIES IS INDICATIVE OF CELLULAR AT AVAILABILITY

Let us return to the cellular environment of bacterial cells. We will first develop a model to show the following conjecture is plausible, i.e., the genomic AT% in rapidly replicating bacterial species can be used as an index of the availability of nucleotides A and T for DNA replication in cellular medium. We will then use this index to (1) study the evolution and adaptation of the bacteriophage genomic AT% in response to the differential nucleotide availability of the host and (2) test the prediction of an association between phage genomic AT% and their host genomic AT%, and (3) test the prediction that double-stranded DNA (dsDNA) phage should exhibit better adaptation than single-stranded DNA (ssDNA) phage. You may be wonder where these predictions come from. Their formulation will be detailed latter.

Designate the amount of the four deoxyribonucleotides dATP, dCTP, dGTP and dTTP available for DNA replication as V_{dA}, V_{dC}, V_{dG} and V_{dT}, respectively. Note that these are abstract terms and may not correspond to the cellular concentration of dNTPs or rNTPs. Suppose a single-stranded DNA genome of length L is composed of A, C, G, and T with frequencies N_A, N_C, N_G and N_T, respectively ($N_A + N_C + N_G + N_T = L$). The polymerization reaction is characterized as

$$M_n \bullet + M \xrightarrow{k_p} M_{n+1} \bullet \tag{4.3}$$

where $M_n\bullet$ stands for an elongating (or propagating in chemistry terminology) DNA strand with n monomer residues (i.e., nucleotides), M is the monomer, and k_p is the propagating constant. According to the law of mass action, and assuming that k_p is the same for adding any of the four nucleotides to the elongating chain, the elongation rate (r) during DNA replication can be modeled as

$$r = \frac{dL}{dt} = k_p [V_{dA}]^{N_A} [V_{dC}]^{N_C} [V_{dG}]^{N_G} [V_{dT}]^{N_T} \qquad (4.4)$$

Bacterial species often need, and typically are selected, to replicate rapidly. For example, *E. coli* in unlimited culture conditions can replicate once every 20 minutes. It is therefore reasonable to assume natural selection to operate on increasing r for such organisms. According to Eq. (4.4), if V_{dA} is the largest, then r is increased with increasing N_A and decreasing N_G, N_C and N_T, with the constraint of $N_A + N_C + N_G + N_T = L$. Without functional constraints such as the genetic code, the maximum r is achieved when $N_A = L$ and $N_C = N_G = N_T = 0$. This means that, in order to maximize r with differential nucleotide availability, the genomic nucleotide usage should evolve to adapt to the availability of nucleotide availability by maximizing the usage of the nucleotide of the highest availability. Similar conclusions have also been derived elsewhere on optimization at the molecular level (Xia, 1996).

One should note that the model above does not consider the effect of differential depletion of the nucleotides. For example, consider that V_{dA} is the largest among the four at the beginning of DNA replication. If a rapidly replicating genome is made entirely of A, then A will be differentially depleted leading to a reduced V_{dA} which consequently may become smaller than V_{dC}, V_{dG} and V_{dT}. This means that the replication of the remaining A-rich part of the genome would be slow, thus compromising the statement above that "The maximum r is achieved when $N_A = L$ and $N_C = N_G = N_T = 0$". However, the qualitative conclusion that, if V_{dA} is larger than V_{dC}, V_{dG} and V_{dT}, then N_A should be larger than N_G, N_C and N_T remains correct.

When $V_{dC} = V_{dG} = V_{dA} = V_{dT} = V$, then Eq. (4.4) becomes:

$$r = \frac{dL}{dt} = kV^{N_A + N_C + N_G + N_T} = kV^L \qquad (4.5)$$

so that r is independent of N_A, N_C, N_G, and N_T. This might be interpreted to mean that, with equal availability of the nucleotides for DNA replication, there is no selection on genomic nucleotide usage and genomic nucleotide frequencies can vary freely. However, the replication of a large, rapidly elongating and AT-rich genome may differentially reduce V_{dC}, V_{dG}, V_{dA}, and V_{dT}. For example, rapid replication of a large AT-rich genome will reduce V_{dA} and V_{dT} and increase the time for adding the remaining A and T to the elongation chain. Thus, even with $V_{dC} = V_{dG} = V_{dA} = V_{dT} = V$ at the beginning of the replication, we would still expect the genomic AT% to be near 50% instead of fluctuating to extreme values.

For a double-stranded genome where $N_A = N_T = N_{AT}$ and $N_C = N_G = N_{CG}$, Eq. (4.4) becomes

$$r = \frac{dL}{dt} = k(V_{dA}V_{dT})^{N_{AT}}(V_{dC}V_{dG})^{N_{CG}} \tag{4.6}$$

If $V_{dA} \cdot V_{dT} >> V_{dC} \cdot V_{dG}$, then increasing N_{AT} in the genome will increase r, with the maximum r achieved when $N_{AT} = L$ and $N_{CG} = 0$, i.e., the genome should evolve towards AT-richness. Again, this assumes no differential depletion of A and T and should be interpreted qualitatively to mean that, with $V_{dA} \cdot V_{dT} >> V_{dC} \cdot V_{dG}$, we should have $N_{AT} > N_{CG}$.

If $V_{dA} \cdot V_{dT} = V_{dC} \cdot V_{dG}$, then r becomes independent of N_{AT} and N_{GC}. However, this again does not necessarily mean that there is no selection to constrain genomic AT% and that genomic AT% can consequently vary freely. As we have argued before, a large, rapidly replicating and AT-rich genome will differentially reduce nucleotides A and T and lead to $V_{dA} \cdot V_{dT} << V_{dC} \cdot V_{dG}$ which is unfavorable for replicating an AT-rich genome. Thus, with $V_{dA} \cdot V_{dT} = V_{dC} \cdot V_{dG}$, we expect the genomic AT% to be near 50% instead of fluctuating to extreme values.

In summary, we expect an extremely GC-rich bacterial genome to indicate relatively high $V_{dC} \cdot V_{dG}$, an extremely AT-rich bacterial genome to indicate relatively high $V_{dA} \cdot V_{dT}$, and a bacterial genome with GC% = 50% to indicate $(V_{dA} \cdot V_{dT}) \approx (V_{dC} \cdot V_{dG})$. It is important to note that, although rATP and dATP cencentrations are generally higher than other rNTPs and dNTPs, this does not preclude the possibility that some bacterial species have greater AT availability than others.

Based on the reasoning above, we may infer that different genomic AT% values in different bacterial species indicate different AT availability in the cells of these bacterial species. By using the genomic AT% of bacterial species as an index of AT availability, we now study how bacteriophage genomic GC% evolve in response to different nucleotide availability in different hosts.

5. FORMULATING THE HYPOTHESIS AND PREDICTIONS

Assuming that it is beneficial for the phage to replicate its genome rapidly, we can make two testable predictions. First, a phage genome should evolve to become AT-rich in a host with a high genomic AT% (indicating $V_{dA}•V_{dT} >> V_{dC}•V_{dG}$ in its cell), and GC-rich in a host with a low genomic AT% (indicating $V_{dA}•V_{dT} << V_{dC}•V_{dG}$ in its cell). This will lead to a positive correlation between the phage genomic AT% and the host genomic AT%. Such a correlation has in fact been known for a long time (Gibbs and Primrose, 1976).

Second, because the rate of spontaneous deamination (Figure 4-2), which leads to C→T or C→U mutations depending on whether C is methylated or not, is about 100-fold higher in the ssDNA than in dsDNA (Frederico *et al.*, 1990), we expect such mutations to reduce the effectiveness of natural selection optimizing the genomic AT% of the ssDNA phage in response to their host genomic AT%. In particular, with low host AT availability, natural selection should favor the reduction of the phage genomic AT%, but the C→T(U) mutation mediated by the spontaneous deamination in the ssDNA phage would counteract against natural selection and increase the genomic AT% of the ssDNA phage.

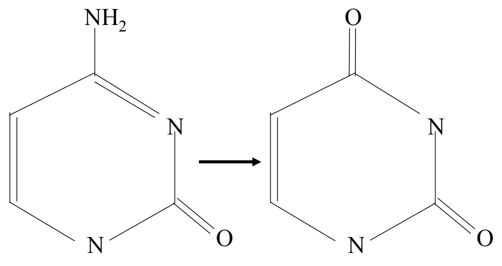


Figure 4-2. Spontaneous deamination converts a cytosine to a uracil.

One more point worth making is that, while A% is constrained to equal T%, and C% equal G%, in dsDNA phage, genomic A% and T% can evolve independently in ssDNA phage. We can therefore specifically predict an increase in the genomic T% in ssDNA phage without an associated increase in the genomic A%. We will test these predictions. There are many complications involved in testing these predictions (Xia and Yuen, 2005). However, we will ignore them at the moment.

6. ARE OUR PREDICTIONS SUPPORTED?

The positive relationship between the phage genomic AT% and their host genomic AT% is shown separately for the dsDNA and ssDNA phages (Figure 4-3). Such a positive relationship itself is trivial because the relationship has been known for nearly 30 years (Gibbs and Primrose, 1976).

However, the difference between the dsDNA and ssDNA phages is scientifically interesting. The regression line for the ssDNA phage has a higher intercept (t = 2.83, p = 0.0028) and a lower slope (t = 2.04, p = 0.0221) than that for the dsDNA phage (Figure 4-3) based on a generalized linear model (Xia and Yuen, 2005).

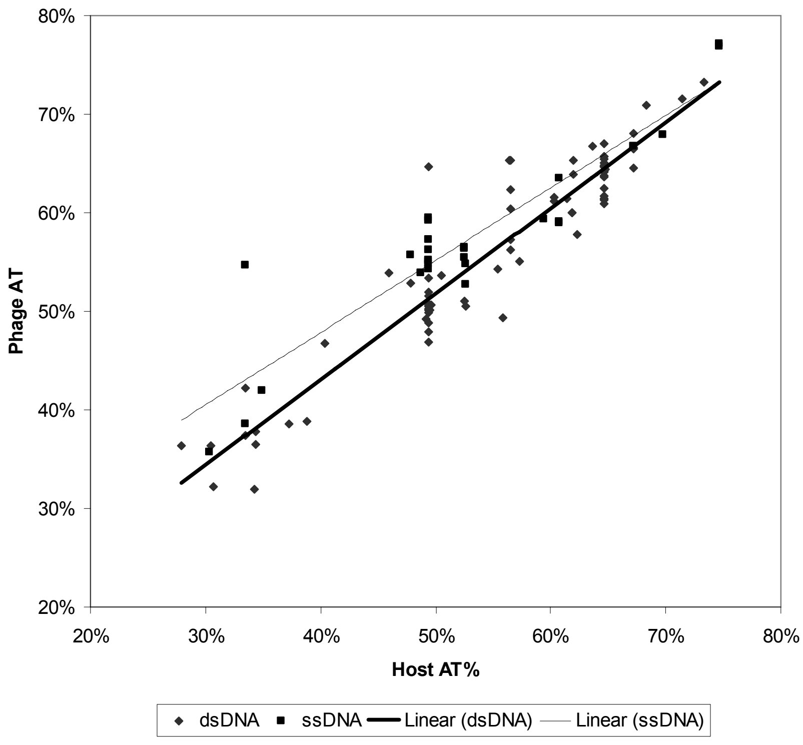


Figure 4-3. Relationship between the phage genomic AT% and the host genomic AT%. Data points for ssDNA and dsDNA phages are plotted separately with their respective linear regression lines.

The increased intercept and decreased slope in the ssDNA phage relative to the dsDNA phage are easy to interpret in light of the finding that the rate of spontaneous deamination, which increases the C→T(U) mutation rate, is about 100-fold higher in ssDNA than in dsDNA (Frederico *et al.*, 1990). This spontaneous deamination features prominently among all other factors contributing to the degradation of DNA (Lindahl, 1993). When host genomic AT% is low (the left extreme of Figure 4-3), indicating low availability of nucleotides A and T in the cellular medium according to equations (4.4) and (4.6), natural selection should cause the phage genome to reduce its AT%, but the C→T(U) mutation mediated by the high rate of spontaneous deamination in the ssDNA phage goes against natural selection and increases phage genomic AT%. In other words, the C→T(U) mutations reduce the effect of the natural selection that pushes the phage genomic AT% downwards. This would raise the intercept and decrease the slope of the regression for the ssDNA phage relative to the regression line for the dsDNA phage.

Note that the C→T(U) mutations act in the same direction as the natural selection when the host genomic AT% is high indicating high availability of nucleotides A and T in the cellular medium according to equations (4.4) and (4.6). In this case, natural selection should favor phage genomes to become AT-rich, and the C→T(U) mutation mediated by the high rate of spontaneous deamination in the ssDNA phage also increases phage AT%, i.e., the two acting in the same direction. Such an interpretation is consistent with the right side of Figure 4-3 in which few points are below the regression line and with little scatter above and below the regression line, especially when the host genomic AT% is extremely high.

To further substantiate this interpretation, we can test whether the increased intercept and decreased slope for the regression line of the ssDNA phage in Figure 4-3 is really due to an increase in the genomic T% instead of the genomic AT%. This can be done because A and T do not need to be equal to each other in number for ssDNA. We expect an increased genomic T% but not genomic A% in the ssDNA phage. Such an inference is consistent with plotting the genomic A% and T% separately for the ssDNA phage against the host AT% (Figure 4-4).

The difference between the two regression lines in Figure 4-4 is significant (Xia and Yuen, 2005). The regression line for the genomic T% has a significantly increased intercept ($P = 0.0068$, one-tailed test) and decreased slope ($P = 0.0323$, one-tailed test). Also, the relationship between the phage genomic A% and the host genomic AT% is stronger than that between the phage genomic T% and the host genomic AT%, with the Pearson correlation coefficient being 0.87857 and 0.60249, respectively.

The results above corroborate our interpretation that C→T(U) mutations contribute significantly to the relationship in nucleotide frequency distribution between the phage genome and the host genome. In particular, the increased intercept and decreased slope for ssDNA phage in Figure 4-4 can be largely attributed to the C→T(U) mutations mediated by the spontaneous deamination.

The pattern in Figure 4-4, however, can have an alternative explanation. First, it is important to note that the host genomic AT% is only indicative of $V_{dA} \bullet V_{dT}$. If V_{dT} is similar in all hosts, but V_{dA} differs substantially among hosts, then $V_{dA} \bullet V_{dT}$ will also differ substantially and phage genomic AT% will consequently be selected to adapt to the host environment of different

V_{dA}•V_{dT}. However, for ssDNA phages in such a scenario with the hosts differing much in V_{dA} but little in V_{dT}, only the genomic A%, but not the genomic T%, of the ssDNA phages will show a good correlation with the host genomic AT%. This is also consistent with the pattern in Figure 4-4.

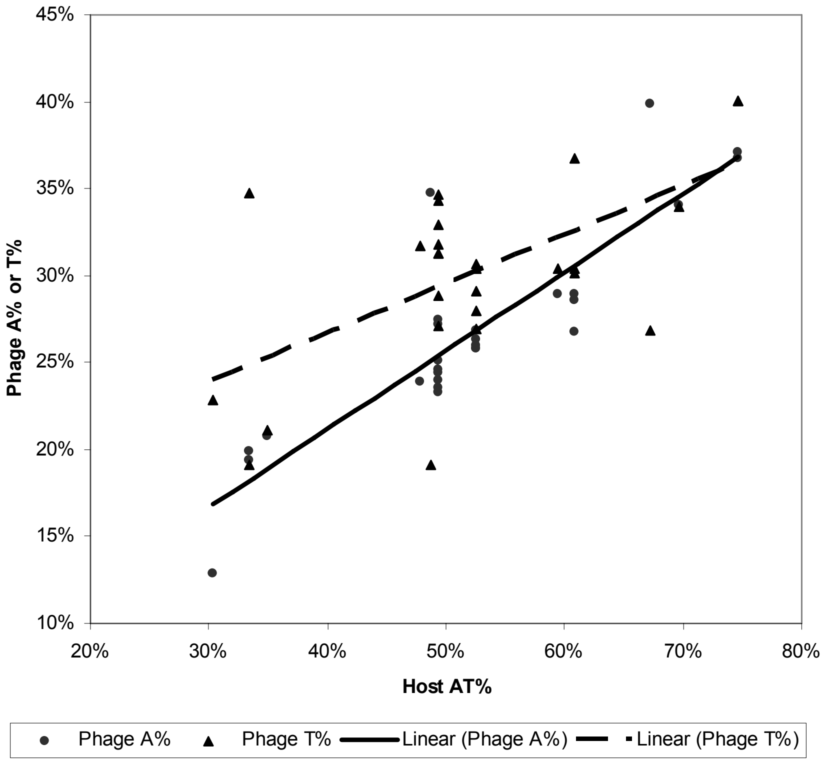


Figure 4-4. The genomic A% and T% of the ssDNA phage plotted against their host genomic AT%. The regression lines are separately fitted for the phage genomic A% and T%, respectively

Mutation and selection are two sculptors of nature, but the effect of mutation on the evolution of genomes in general and proteins in particular is only recently appreciated (Gu *et al.*, 1998; Hickey and Singer, 2004; Lobry, 2004; Wang *et al.*, 2004), notably after the pioneering work of Sueoka (1961). The C→T(U) mutations mediated by spontaneous deamination (Frederico *et al.*, 1990, 1993; Lindahl, 1993; Sancar and Sancar, 1988), in particular, have been invoked to explain the strand-asymmetry in nucleotide frequency distribution in vertebrate mitochondrial genomes (Reyes *et al.*, 1998; Tanaka and Ozawa, 1994; Xia, 2005c), in the bacterial genomes

(Lobry, 1996; Lobry and Sueoka, 2002; McInerney, 1998), and in coding sequences (Beletskii and Bhagwat, 1996, 1998, 2001; Beletskii *et al.*, 2000). The result presented above shows how the C$\rightarrow$T(U) mutations can operate together with selection to shape the genomic AT% of dsDNA phage and the genomic A% and T% in ssDNA phage.

Previous studies have shown that spontaneous mutation appears to be AT-biased in different genomes and genetic backgrounds (Gojobori *et al.*, 1982; Li *et al.*, 1984; Marcelino *et al.*, 1998; Wang *et al.*, 1996), and the evidence is convincing based on the comparison between functional genes and their pseudogene counterparts (Gojobori *et al.*, 1982; Li *et al.*, 1984). However, mutation alone is often insufficient to explain the observed genetic variation.

Two different kinds of AT-richness have been documented for mitochondrial genomes alone demanding two different explanations (Xia, 1996). The first kind is represented by (1) the insect mitochondrial genomes where most codons end with A and T and (2) the mammalian mitochondrial D-loop which is not transcribed and very AT-rich. Both the D-loop and the third codon position of protein-coding genes evolve rapidly. In the insect mitochondrial genomes, the number of A-ending codons roughly equals the number of T-ending codons. In the D-loop, the number of A and T are distributed roughly equally in the two strands. This first kind of AT-richness was attributed to AT-biased mutation (Xia, 1996). The second kind of AT-richness is represented by the coding sequences in vertebrate mitochondrial genomes, where most codons in four-fold degenerate codon families end with A but few end with T. This cannot be explained by the mutation hypothesis invoking AT-biased mutation because such mutations would lead to roughly equal number of A-ending and T-ending codons in four-fold degenerate codon families.

The large number of A-ending codons with few T-ending codons in mammalian mitochondrial genomes prompted the proposal of the transcription hypothesis of codon usage (Xia, 1996), based on the observation that cellular concentration of ATP is much higher than that of the other three rNTPs (Table 2.1 in Bridger and Henderson, 1983, pp. 4-5; Colby and Edlin, 1970; Table 2.1 in Kornberg and Baker, 1992). For example, in the exponentially proliferating chick embryo fibroblasts in culture, the concentration of rATP, rCTP, rGTP and rUTP, in units of (moles $\times 10^{-12}$ per 10^6 cells), is 1890, 53, 190, and 130, respectively, in 2-hour culture, and 2390, 73, 220, and 180, respectively, in 12-hour culture. The transcription hypothesis of codon usage states that, with the high availability of rATP and relatively low availability of the other three rNTPs, especially rCTP, the transcription efficiency can be increased by maximizing the use of A in the third codon position of protein-coding genes. The limitation of rCTP is well-exemplified by the protozoan parasite, *Trypanosoma brucei*, in mammalian blood. The parasite maintains its *de novo* synthesis pathway for CTP and inhibiting its CTP synthetase effectively eradicates the parasite population in the host (Hofer *et al.*, 2001). This suggests that little CTP can be salvaged from the host. In contrast, the parasite does not have *de novo* synthesis pathways for purines, suggesting that the parasite can obtain the purines by its salvage pathway. C-limitation appears to be a general feature in bacterial species, and a biochemical explanation has been offered to explain the general C-limitation in bacterial species (Rocha and Danchin, 2002). C-limitation has also been invoked to explain the length of coding sequences (CDSs), i.e., a long CDS with many Cs may take inordinately long to be transcribed and should therefore be selected against (Xia *et al.*, 2006).

The variation of the genomic AT% in the dsDNA phage and the genomic A% and T% in the ssDNA phage in our study cannot be explained by the C→T(U) mutations alone, and we believe that the correlations shown in Figure 4-3 and Figure 4-4 are mainly the work of natural selection favoring the AT-rich phage in AT-rich hosts and AT-poor phage in AT-poor hosts. The data from ssDNA phage helped us to conclude that it is the C→T(U) mutations, instead of AT-biased mutations, are mainly responsible for the difference between the ssDNA and dsDNA phages we observe in Figure 4-3 and Figure 4-4. The results here corroborate a previous finding (Xia, 2005c) that spontaneous deamination has profound effect on the strand-biased nucleotide and codon frequency distributions and on the codon-anticodon adaptation in another kind of intracellular genomes, i.e., the vertebrate mitochondrial genomes.

In short, the phage genomic AT% has evolved in response to the availability of A and T in their host cells. In particular, the difference in the relationship between the ssDNA phage and dsDNA phage, can be partially explained by the difference in (1) selection operating to maximize the rate of DNA replication and (2) the C→T(U) mutation mediated by the high rate of spontaneous mutations in the ssDNA phage.

The key message from this chapter is that researchers in bioinformatics need to gain a certain degree of familiarity with biology, especially molecular biology and microbiology. The next three chapters bring us back to core bioinformatics tools, but Chapter 8 will again expose us to more fundamentals of biology, especially on the three essential biological processes: DNA replication, transcription and translation.

7. A SHORT PLAY FEATURING PHAGES AND BACTERIA

(Phage virions advancing towards bacterial cells)

Bacteria: Hey. Stop! We are peace-loving creatures.

Phages: But we see restriction enzymes made inside your cells with a clear purpose of cutting us into pieces. Such weapons of mass destruction cannot be tolerated, and we have been called upon to come in and kill you.

Bacteria: But look, the restriction enzymes we make are called 8-cutters. They have a recognition site with 8 nucleotides. The chance of having such a lengthy recognition site found in your tiny little genome is extremely small. In fact, if you BLAST this recognition site against your genome, you will find absolutely no match. On the other hand, you will find several matches of the restriction site against our own larger genomes. This shows that the restriction enzyme is really for maintaining our own genomic integrity under special circumstances.

Phages: Glad to know that your restriction site does not have any match against our genome, which suggests the right timing for our preemptive strikes. It would have been too late if you already have a restriction enzyme that can find a cutting site in our genome.

(So phages invade the bacterial cells, proliferate and finally vote to destroy the bacterial cells. Remaining bacterial cells all shrink into a corner and look hostile towards the advancing phages.)

Phages: Be nice to us, otherwise we will come to vote inside you, too!

(Just in case you do not know, the production of restriction enzymes in bacterial hosts imposes strong selection against their respective phage to such an extent that almost all phage genomes exhibit strong avoidance of the restriction sites. You may take an *Escherichia coli* or *Bacillus subtilis* genome to verify this point quite easily.)

Chapter 5

GENE AND MOTIF PREDICTION

1. INTRODUCTION

Two major categories of gene and motif *ab initio* annotation methods are in current use. The first is based on known genes in molecular databases, and uses homology search tools such as FASTA (Pearson and Lipman, 1988) and BLAST (Altschul *et al.*, 1990; Altschul *et al.*, 1997), which are the subject of Chapter 1. The second, better known as gene and motif prediction, is based on known gene structures such as exon-intron structures in eukaryotic protein-coding genes, and represented by GENSCAN (Burge and Karlin, 1997). It might be of interest to note that both Stephen F. Altschul and Chris Burge published their respective works on BLAST and GENSCAN when they were in the research group of Samuel Karlin in Department of Mathematics, Stanford University.

While some gene and motif prediction methods, such as Gibbs sampler that will be covered in a latter chapter, do not require known differences between coding and non-coding sequences or between motifs and non-motifs, most gene prediction methods are based on known sequence differences between protein-coding sequences and non-coding sequences and between motifs and non-motifs. The differences are often characterized into two categories, the signal sensors and content sensors. The signal sensor refers to signals with strong site dependence, i.e., knowing a nucleotide, an amino acid or a word (e.g., a nucleotide triplet or an amino acid doublet) at site i greatly improves our prediction of the nucleotide, amino acid or word at site i+k (where k is any integer). For example, a long word called anti-Shine-Dalgarno sequence (Shine and Dalgarno, 1975b) at site i in bacterial mRNA greatly improves our prediction of the presence of an ATG triplet about 10 bases downstream. The content sensor, on the other hand, is not supposed to have detectable site dependence. For example, in a long intron or in a DNA sequence that has never been transcribed during its evolutionary history, knowing a nucleotide at site i generally will not improve our prediction of what nucleotide should be found at site i+k. However, even for such a sequence, its nucleotide, dinucleotide or trinucleotide frequencies (generally referred to as word frequencies) may help us to know whether it is coding or non-coding. These word frequencies are typical example of content sensors.

Extensive studies have been carried out to characterize the differences among different sequence states (e.g., transcription start and termination sites, translation initiation and termination sites, exons, introns, poly-A site, etc.), leading to a variety of signal sensors such as the relatively uniform splicing sites (Burge, 1998; Foissac and Schiex, 2005; Gelfand, 1989; Tenney *et al.*, 2004) and the much less uniform exon-exon junctions in spliced mRNA (Gelfand, 1992), and content sensors such as unusual frequency distributions of words (Borodovsky and McIninch, 1993b; Gelfand *et al.*, 1992; Pevzner *et al.*, 1989) that can be potentially used in gene-finding. All these pieces of information can be combined in a Bayesian framework to increase the confidence of gene prediction. We will have a very gentle exposure to Bayes' theorem in this chapter and Bayesian inference in Chapters 13 and 14.

In this chapter we will first learn a few commonly used techniques for characterizing the features of signal sensors and content sensors. This will be followed, in the next chapter, by an in-depth account of hidden Markov models that combine information from both the signal sensors and content sensors. Before we get into serious bioinformatic tools, please allow me to smuggle into the book some basic terminology involved in informal decision making.

2. BAYES' THEOREM AND ODDS RATIOS

With N alternative and discrete hypotheses individually designated as θ_i (where i = 1, 2, …, N), the observation Y and the prior probabilities associated with each hypothesis as $P(\theta_i)$, Bayes' theorem expresses the probability of θ_i being true, given the observed Y, as

$$P(\theta_i \mid Y) = \frac{P(Y \mid \theta_i)P(\theta_i)}{\sum_{j=1}^{N} P(Y \mid \theta_j)P(\theta_j)} \tag{5.1}$$

When there are only two alternative hypotheses, the theorem is reduced to the familiar form in basic statistical books:

$$P(\theta_i \mid Y) = \frac{P(Y \mid \theta_i)P(\theta_i)}{P(Y \mid \theta_1)P(\theta_1) + P(Y \mid \theta_2)P(\theta_2)} \tag{5.2}$$

$P(\theta_i|Y)$ is the posterior probability for hypothesis θ_i, $P(Y|\theta_i)$ is the likelihood defined as the conditional probability of having the observation Y given hypothesis θ_i, and $P(\theta_i)$ is the prior probability. The numerator and denominator are know as the joint and marginal probabilities, respectively.

Suppose we believe that an average reader of this book has a probability of 0.6 of university-level mathematical training. This naturally means that 40% or readers are believed not to have the training. We also have some sampling data to show that those with the university-level mathematics training will have a probability of 0.9 of getting as far as this chapter, and those without will have a probability of 0.2 of getting to this chapter. Now suppose a reader has come to this chapter. What is the probability that the reader actually has university-level mathematical training?

This problem, clearly involving conditional probabilities, can be easily solved by applying Bayes' theorem. We have two hypotheses, i.e., either the reader has the university-level mathematical training, defined as $\theta_{Yes,}$ or he does not, defined as θ_{No}. We also define the observation of "reaching this chapter" as Y. Before we have this observation, our best guess of the reader having university-level training is 0.6. This leads to the specification of the two prior probabilities:

$$\begin{aligned} &P(\theta_{Yes}) = 0.6 \\ &P(\theta_{No}) = 1 - P(\theta_{Yes}) = 0.4 \end{aligned} \tag{5.3}$$

Because our sampling data show that a reader has a probability of 0.9 of getting to this chapter if he has university-level math training and a chance of 0.2 if he does not, we can now specify the two likelihoods:

$$\begin{aligned} &L_{Yes} = P(Y \mid \theta_{Yes}) = 0.9 \\ &L_{No} = P(Y \mid \theta_{No}) = 0.2 \end{aligned} \tag{5.4}$$

Now the probability of the reader having university-level training (the posterior probability), given the observation that he has reached this chapter, is simply

$$P(\theta_{Yes} \mid Y) = \frac{0.9 \times 0.6}{0.9 \times 0.6 + 0.2 \times 0.4} = 0.871 \tag{5.5}$$

The knowledge that the reader has reached this chapter has changed the prior probability of 0.6 to 0.871. The probability that θ_{No} is true is naturally 0.129.

It is important to keep in mind the fact that, if a prior probability is zero, then the associated posterior probability is also zero. For example, if $P(\theta_{Yes})$ is zero, then $P(\theta_{Yes}|Y)$ is also zero, and no evidence to the contrary will ever change it. Thus, a scientist has an obligation not to take extreme views when adopting a Bayesian approach. Setting a prior probability to extreme values has done much harm in world politics. Setting the prior probability of a certain country having WMD to 1 will always lead to a posterior probability of 1, and no evidence to the contrary will ever change it.

We now introduce an index to measure how likely a hypothesis is relative to its alternative: the odds ratio which is defined as the ratio of the two probabilities each associated with a hypothesis. For example, the odds ratio of the two likelihood values in Eq. (5.4), also known as the likelihood ratio, is

$$\Omega = \frac{P(Y \mid \theta_{Yes})}{P(Y \mid \theta_{No})} = \frac{0.9}{0.2} = 4.5 \tag{5.6}$$

Classical statistical decision making is based solely on the observation and nothing else. So in this case, given the observation of the reader having reached this chapter, θ_{yes} is 4.5 times as likely as θ_{No}. On the other hand, if we do not know that the reader has reached this point, then we only have the odds ratio of the two prior probabilities

$$\Omega' = \frac{P(\theta_{Yes})}{P(\theta_{No})} = \frac{0.6}{0.4} = 1.5 \tag{5.7}$$

The two posterior probabilities combine information from both our prior knowledge (expressed in prior probabilities) and our new observation and yield the following odds ratio

$$\Omega'' = \frac{P(\theta_{Yes} \mid Y)}{P(\theta_{No} \mid Y)} = \frac{0.871}{0.129} = 6.75 \tag{5.8}$$

You may already know that Bayes' theorem can also be expressed as

$$\Omega'' = \Omega'\Omega \tag{5.9}$$

You can numerically verify the correctness of this formulation by using the three values in Eq. (5.6)-(5.8). This expression shows that the odds ratio of the posterior probabilities is equal to the likelihood ratio multiplied by the odds ratio of the prior probabilities

Let us now have a more relevant, but also a bit more complicated, example. Suppose that a bacterial genome of length L contains N protein-coding genes of length X_1, X_2, …, X_N. Suppose we randomly pick up a sequence fragment of length z bases, the probability that it is within a coding sequence is

$$p = \frac{\sum_{i=1, X_i \geq z}^{N} (X_i - z + 1)}{L - z + 1} \tag{5.10}$$

where the numerator is the number of possible ways of landing the segment fragment of z bases long within a coding sequence and the denominator is the number of possible ways of landing the fragment of z bases within the genome. For numerical illustration, let's fix z = 90 bases and work with a linear genome that is so simple that it has a length of only 10,000 bases and contains only two protein-coding genes of lengths 900 bases and 3000 bases, respectively, and three non-coding sequences, with one between the two coding genes, one at the 5'-end and the other at the 3'-end of the genome. This gives

$$p = \frac{(900 - 90 + 1) + (3000 - 90 + 1)}{10000 - 90 + 1} = 0.37554 \tag{5.11}$$

Just in case that you are not a biology major, we define an open reading frame (ORF) as a series of consecutive sense codons flanked by an inframe upstream initiation codon (e.g., ATG) and an inframe downstream termination codon (e.g., TAA). By analyzing a set of known protein-coding genes and intergenic sequences from the genome, you found that the probability of a protein-coding gene having an ORF at least 90 bases long is 0.95 and the probability of an intergenic sequence having an ORF at least 90 bases long is only 0.1. Now if you have a sequence of 90 bases long that happens to be an ORF, what is the probability that it is a coding sequence?

Again we have two hypotheses, i.e., the 90-base segment is a protein-coding gene (θ_{Yes}) or it is not (θ_{No}). Our prior probabilities are

$$\begin{aligned} P(\theta_{Yes}) &= 0.37554 \\ P(\theta_{No}) &= 0.62446 \end{aligned} \tag{5.12}$$

The likelihood values associated with the two hypotheses, given the observation that the 90-base segment is an ORF is

$$\begin{aligned} P(Y \mid \theta_{Yes}) &= 0.95 \\ P(Y \mid \theta_{No}) &= 0.1 \end{aligned} \tag{5.13}$$

The probability that θ_{Yes} is correct is then

$$P(\theta_{Yes} \mid Y) = \frac{0.95 \times 0.37554}{0.95 \times 0.37554 + 0.1 \times 0.62446} = 0.851 \tag{5.14}$$

Thus, knowing the 90-base segment is an ORF greatly increased the chance that it is a coding sequence (from the prior probability of 0.37554 to the posterior probability of 0.851). The concept of odds ratios will be used in position weight matrix in the next section and latter chapters.

3. CHARACTERIZING FEATURES OF SIGNAL SENSOR

3.1 Position weight matrix

Position weight matrix (PWM) is a simple technique for characterizing sequence motifs from a set of aligned training sequences. The resulting PWM can be used to scan sequence fragments and generate a PWM score for each sequence fragment, with a large score associated with a higher likelihood of the fragment being one of the motifs. PWM is not only quite useful in its own right, but is also an essential building block in Gibbs sampler used often in detecting regulatory sequences in DNA or functional motifs in proteins. We will cover Gibbs sampler for motif prediction in a latter chapter. Here we will first describe the details of computing a PWM and obtaining a PWM score for each sequence, and then highlight a few of its limitations.

Suppose we want to characterize the translation initiation signal in eukaryotic mRNAs. According to Kozak's scanning model (Kozak, 1982, 1984, 1989), the translation initiation signal in eukaryotic mRNAs includes the initiation codon together with a few bases flanking the initiation codon. So we may take, say five bases, flanking the initiation codon from each known mRNAs as sequence input (i.e., training sequences) to generate PWM.

In this illustrative example, we will use only protein-coding genes on human chromosome 22, which is the shortest and first sequenced human autosome (Dunham *et al.*, 1999). The chromosome contains 508 annotated protein-coding genes with real names, i.e., not genes named as LOC###### where "#' is an integer number. The raw sequence data are partially displayed below, with the left column being the gene name:

```
A4GALT       ATACCATGTCCAA
ACO2         ACAAAATGGCGCC
ACR          GGAGTATGGTTGA
ADM2         CCGCCATGGCCCG
......
```

At this point we hardly see (unless you already have trained eyes) any similarity among the 13-base sequences other than the fact that they all have the initiating ATG codon in the middle with 5 bases flanking on each side.

The first step in generating PWM is to obtain site-specific nucleotide frequencies (or amino acid frequencies if we are characterizing a sequence motif shared among proteins in a protein family). If the motif has extremely biased nucleotide usage, then that would provide a straightforward way of predicting such a motif. However, the result of nucleotide frequencies (Table 5-1) does not show particularly biased nucleotide usage except that the three sites occupied by the initiation codon.

Designate f_{ij} as the site-specific frequency for nucleotide i (i = 1, 2, 3, 4 corresponding to A, C, G, and T) at site j (=1, 2, …, 13 in our example in Table 5-1), and the number of sequences as N, the probability of encountering nucleotide i at site j is estimated as

$$p_{ij} = \frac{f_{ij}}{N}, \text{ e.g.,} \qquad \qquad (5.15)$$
$$p_{A1} = \frac{75}{508} = 0.1476$$

In contrast, the probability of encountering nucleotide i without any site information is estimated as

$$p_i = \frac{\sum_j^L f_{ij}}{N \bullet L}, \ e.g., \qquad (5.16)$$
$$p_A = \frac{1563}{508 \times 13} = 0.2367$$

where L is sequence length.

Table 5-1. Site-specific nucleotide frequencies from the 508 named protein-coding genes on human chromosome 22. The initiation codon ATG is located at sites 6-8.

Site	A	C	G	U
1	75	173	171	89
2	105	216	144	43
3	199	70	212	27
4	124	236	83	65
5	60	276	143	29
6	502	6	0	0
7	0	0	0	508
8	0	0	508	0
9	98	71	274	65
10	141	227	89	51
11	49	154	221	84
12	89	144	196	79
13	121	151	134	102
Sum	1563	1724	2175	1142

Given a sequence, say, S = ACGGTACCACGTT, we have two hypotheses, i.e., it belongs to the 13-base translation initiation signal (θ_{Yes}) or it does not (θ_{No}). It should share the site dependence with the training sequences if θ_{Yes} is true, and not if θ_{No} is true. In the latter, it does not. The likelihoods of observing sequence S, given the two different hypotheses, are specified, respectively, as

$$L_{Yes} = p(S \mid \theta_{Yes}) = p_{A1} p_{C2} p_{G3} p_{G4} \cdots p_{T13}$$
$$L_{No} = p(S \mid \theta_{No}) = p_A^3 p_C^4 p_G^3 p_T^3 \qquad (5.17)$$

In statistical inference, you will often encounter notations equivalent to p(S|θ_{Yes}) or p(S|θ_{No}). You should instantly recognize it as a likelihood function. You may note that, for P(S|θ_{No}), the order of sites is irrelevant.

$P(S|\theta_{No})$ remains the same if you rearrange the nucleotides in sequence S in any order. For $P(S|\theta_{Yes})$, the order is important and rearrangement of the nucleotides in S will change $P(S|\theta_{Yes})$.

If $P(S|\theta_{Yes})$ is much larger than $P(S|\theta_{No})$, then we tend to consider S as a member of the 13-base translation initiation sequence, especially when the initiation codon is excluded from computation. One convenient index would seem to be the odds ratio of $P(S|\theta_{Yes})/P(S|\theta_{No})$, which you should recognize as the likelihood ratio. The problem is that both $P(S|\theta_{Yes})$ and $P(S|\theta_{No})$ would become very small when S has even a moderate length. For example, assuming equal nucleotide frequencies, $P(S|\theta_{No}) = 0.25^{13} = 0.000000015$. Computation involving small numbers is always plagued by rounding errors and overflows. For this reason, it is not the odds ratio but the logarithm of the odds ratio (or log-odds for short) that is used as a measure of how likely that the sequence belongs to the 13-base translation initiation site. This log-odds (or log-likelihood ratio) is called the sequence score of the position weight matrix (PWMS):

$$PWMS = \log_2\left(\frac{p(S\,|\,\theta_{Yes})}{p(S\,|\,\theta_{No})}\right) = \log_2\frac{p_{s_1 1}}{p_{s_1}} + \log_2\frac{p_{s_2 2}}{p_{s_2}} + \ldots + \log_2\frac{p_{s_L L}}{p_{s_L}} \quad (5.18)$$

Using the logarithm of base 2 makes it easy to see the difference between $P(S|\theta_{Yes})$ and $P(S|\theta_{No})$, i.e., PWMS = 1 when the ratio is 2, PWMS = 2 when the ratio is 4, etc. One does not have to use base 2 and there is no agreed-upon convention for choosing any base. Also, sometimes likelihood ratio, instead of its logarithm, is used as PWMS.

The PWM is a matrix to facilitate the computation of PWMS. It is of the same dimensions as the site-specific frequency matrix, i.e., 4 by 13 in our case. The site-specific frequency matrix in Table 5-1 is presented as a 13 by 4 matrix, instead of a 4 by 13 matrix, because of the limitation of page width. Each individual entry in PWM is

$$PWM_{ij} = \log_2\left(\frac{p_{i,j}}{p_i}\right) \quad (5.19)$$

In practice, PWM_{ij} is computed with the following equation which is computationally more efficient than Eq. (5.19):

$$PWM_{i,j} = \log_2 L + \log_2\left(\frac{f_{i,j}}{f_i}\right) \quad (5.20)$$

where L is the sequence length (13 in our case), i = 1, 2, 3 and 4 corresponding to A, C, G and T, j = 1, 2, …, 13 indicating the site number, $f_{i,j}$ is the nucleotide frequency of nucleotide i at site j, e.g., $f_{A,1} = 75$ (Table 5-1) and f_i is the overall nucleotide frequency of nucleotide i, e.g., $f_A = 1563$ (Table 5-1). One may also use the genomic nucleotide frequencies for f_i, especially when the nucleotide frequencies are highly biased in the motif relative to the genomic frequencies.

A positive PWM_{ij} value means it is more likely, and a negative PWM_{ij} value means it is less likely, to find nucleotide i at position j than random expectation based on p_i values only. A PWM_{ij} value of 0 means that finding nucleotide i at position j is not informative in discriminating between motifs and non-motifs.

Note that f_{ij} may be zero, e.g., $f_{G6} = f_{T6} = f_{A7} = f_{C7} = f_{G7} = f_{A8} = f_{C8} = f_{T8} = 0$ in Table 5-1, and no logarithm is defined for zero. One common approach to avoid this problem is to add what is called pseudocounts. Define α as a scaling variable for computing pseudocounts, we have

$$\begin{aligned} f_{i.pseudo} &= \alpha f_i \\ f_{pseudo} &= \sum_{i=1}^{N} f_{i.pseudo} \end{aligned} \tag{5.21}$$

where i = 1, 2, 3, 4 corresponding to A, C, G, and T or 1, 2, …, 20 corresponding to the 20 amino acids, respectively, and N = 4 for nucleotide sequences or 20 for amino acid sequences. Now we modify Eq. (5.20) to take the following form:

$$PWM_{i,j} = \log_2 L + \log_2\left(\frac{f_{i,j} + f_{i.pseudo}}{f_i + f_{pseudo}}\right) \tag{5.22}$$

You may wonder how to choose the α value. The allowable values are between 0 and 1, and some programs such as Gibbs Motif Sampler (Thompson *et al.*, 2004; Thompson *et al.*, 2003) have a default value of 0.1. Strangely enough, I have never seen any rules proposed to govern the choice of the α value. So I will add a few sentences on this topic.

It is important to know the consequences choosing different α values because it affects the PWM score (PWMS) that we need to calculate for each sequence. If α = 0, then PWMS is expected to be 0 for a sequence randomly generated from the pool of background nucleotide frequencies (i.e., p_A, p_C, p_G, p_T). With α > 0, PWMS is no longer expected to be 0 and the interpretation of its absolute value becomes impossible. For this reason, to paraphrase Albert Einstein, the α value should be as small as possible, but not smaller. I recommend 0.01 because such a small α value has little effect on PWMS. The default α value in DAMBE (Xia, 2001; Xia and Xie, 2001b) is 0.01.

The PWM obtained with Eq. (5.22) based on all the 13-base translation initiation site from the annotated human chromosome 22 sequences is shown in Table 5-2, with α = 0.05. The largest values are found at sites 6-8 corresponding to ATG in the middle of the 13-mer. Highlighting the largest values in each row flanking ATG results in the recovery of Kozak's translation initiation consensus, i.e., RCCATGG (Table 5-2). In translation literature, A in ATG is often designated as site 1, and the first R is referred to as -3R and the last G as +4G. Advocates of the scanning model often attribute much significance to these two sites as important signals for recognition of translation initiation, so does nearly every textbook on molecular biology.

Table 5-2. PWM derived from the site-specific frequencies in Table 5-1, using Eq. (5.22) and with the last row of Table 5-1 as f_i values. Pseudocounts equal to 5% of the actual frequencies were used. The Kozak translation initiation consensus is in bold. PWM values computed with DAMBE (Xia, 2001; Xia and Xie, 2001b). The standard deviation in the last column (Std) is an indication of the amount of information at each site and is highly correlated with Shannon's information entropy (Shannon, 1948).

Site	A	C	G	U	Std
1	0.0726	0.7140	0.5377	0.3675	0.2731
2	0.3307	0.9354	0.3913	-0.1780	0.4553
3	**0.9284**	-0.0167	**0.7350**	-0.4293	0.6367
4	0.4731	**1.0279**	-0.0072	0.1086	0.4656
5	-0.0761	**1.1967**	0.3856	-0.3954	0.6915
6	**1.9941**	-0.7772	-0.8254	-0.9879	1.4316
7	-0.8980	-0.8743	-0.8254	**2.3190**	1.5927
8	-0.8980	-0.8743	**1.6783**	-0.9879	1.3001
9	0.2745	-0.0075	**0.9900**	0.1086	0.4476
10	0.5896	0.9870	0.0373	-0.0671	0.4931
11	-0.1958	0.6042	0.7750	0.3173	0.4249
12	0.1988	0.5428	0.6612	0.2652	0.2207
13	0.4515	0.5860	0.3331	0.4905	0.1047

It is important to keep in mind that a consensus sequence obtained by the PWM method does not mean that it has anything to do with translation initiation. In fact, the interpretation of +4G has been controversial. It has been suggested that +4G may have little to do with initiation site recognition, but is constrained by the requirement for particular type of amino acid residue at the N-terminus of the protein (Cigan and Donahue, 1987). One piece of supporting evidence came from a detailed study of an influenza virus NS cDNA derivative (Grunert and Jackson, 1994) which showed that both +4 and +5 sites were important and changes at these sites reduced protein production. In contrast, the +6 site (the third codon position of the second codon) is less important. A simple explanation of this result is that changes at the +4 and +5 sites alter the amino acid, whereas those at the +6 site may not.

Recent studies, especially those involving the removal of the initiator methionine (Met) and myristoylation, revived the alternative explanation of amino acid constraint for the presence of +4G in protein-coding genes. First, amino-terminal modifications of nascent peptides occur in nearly all proteins in both prokaryotes and eukaryotes, and the removal of the initiator Met, which occurs soon after the amino terminus of the growing polypeptide chain emerges from the ribosome, is not only an important amino-terminal modification in itself, but also required for further amino-terminal modifications. The efficiency of removing the initiator Met depends heavily on the penultimate (the second) amino acid, with the efficient cleavage occurring only when the penultimate amino acid is small (Moerschell *et al.*, 1990). Alanine (Ala) and glycine (Gly) happen to be the two smallest amino acids and both are coded by G-starting codons, i.e., Ala by the GCN (where N stands for any nucleotide) and Gly by the GGN codons. The need for removing the initiator Met in proteins implies the presence of many Ala and Gly at the penultimate amino acid position and consequently many +4G due to the GCN and GGN codons coding for Ala and Gly, respectively.

Another factor contributing to the prevalence of +4G, but independent of the efficiency of translation initiation, is the myristoylation process. For example, in Coxsackievirus B3, the initiation codon is flanked by both -3R and +4G, and viral mutants with a mutation from +4G to +4C is not viable (Harkins *et al.*, 2005). This may seem to confirm what one would expect based on the necessity of the Kozak consensus for efficient translation initiation in highly expressed genes. However, it turns out that the +4G is required in Coxsackievirus B3 not because it is essential for translation initiation, but because it is needed for coding Gly (coded by GGN). The Gly at the amino terminus, after the removal of the initiator methionine, is needed to attach to a myristoyl ($C_{14}H_{28}O_2$) fatty acid side chain, and myristoylation occurs only on a Gly residue (Farazi *et al.*, 2001). Myristoylation may involve many proteins, and are implicated in protein subcellular relocalization (Farazi *et al.*, 2001), apoptosis (Sakurai and Utsumi, 2006; Vilas *et al.*, 2006), signal transduction (de Vries *et al.*, 2006; Rowe *et al.*, 2006), and the virulence and colonization of pathogens (Bentham *et al.*, 2006; Breuer *et al.*, 2006; Harkins *et al.*, 2005; Provitera *et al.*, 2006; Robert-Seilaniantz *et al.*, 2006). The need for myristoylation in proteins would contribute to the presence of +4G in CDSs.

We thus have two alternative hypotheses for the presence of +4G in protein-coding genes. The conventional translation initiation hypothesis argues that the presence of +4G is necessary for highly expressed proteins, with two predictions. First, the selection favoring +4G should drive the increased usage of amino acids coded by GNN codons (e.g., Ala coded by GCN, Asp by GAY, Glu by GAR, Gly by GGN, and Val by GUN) at the penultimate amino acid site. Second, the +4G should be more prevalent in highly expressed than in lowly expressed genes. In contrast, the amino acid constraint hypothesis, based on the amino-terminal modification involving the removal of the initiator Met and myristoylation, has two different predictions. First, not all GNN codons should have increased usage, but only GCN coding Ala and GGN coding Gly should have increased usage. Second, highly expressed genes may need more efficient N-terminal processing and may consequently need more GCN and GGN codons. This may increase the frequency of +4G in highly expressed genes relative to lowly expressed genes.

To those two alternative hypotheses, one could add still one more. It is known that translation initiation is often the rate-limiting step during protein production (Bulmer, 1991; Liljenstrom and von Heijne, 1987), and it is advantageous fro the ribosome to move quickly down the translation initiation site. Because tRNA^{Ala} consistently has more gene copies in both eukaryotes and prokaryotes, alanine codons are expected to be translated faster than other codons. For this reason, highly expressed genes should have alanine codons right after the initiation codon so that ribosome can quickly move downstream to clear the translation initiation site. This hypothesis, which may be termed tRNA hypothesis of translation initiation, has unique predictions. First, it predicts only alanine codons (GCN) should be the greatest contributor to +4G, and this pattern should be stronger in highly expressed genes than lowly expressed genes. Second, it predicts that any species, being it prokaryotes or eukaryotes, should feature +4G regardless of how translation is initiated, as long as tRNAs for amino acids coded by GNN codons are the most abundant.

Let us get back to PWM. In artificial intelligence literature, the process of obtaining the PWM is called training, and the application of the PWM to classify unknown sequence fragments as containing or not containing a translation initiation signal is called prediction. Suppose we want to use the PWM in Table 5-2 for prediction. What we will do is to scan each 13-mer along an unknown sequence (S) and compute a PWM scores (PWMS) for each 13-mer. For example, a sequence S = CCACCATGGCTGTG….. will have the following overlapping 13-mer's, starting at positions 1, 2, …:

```
CCACCATGGCTGT
CACCATGGCTGTG
ACCATGGCTGTG.
......
```

Once PWM is computed, PWMS for each 13-mer is computed as

$$PWMS = \sum_{j=1}^{L} PWM_{S_j,j} \tag{5.23}$$

where L is the length of the sequence fragment (= 13 in our case) and PWM values are from Table 5-2. For the first fragment, i.e., CCACCATGGCTGT,

$$\begin{aligned} PWMS &= PWM_{C,1} + PWM_{C,2} + PWM_{A,3} + ... + PWM_{T,13} \\ &= 0.7140 + 0.9354 + 0.9284 + ... + 0.4905 = 14.2398 \end{aligned} \tag{5.24}$$

The neutrophil cytosolic factor 4 (NCF4) gene happens to have its translation initiation sequence identical to CCACCATGGCTGT, so its PWMS is 14.2398. In contrast, the sequence flanking the initiation codon in the IGLVIV-59 gene, which is a pseudogene, is TTGTGCTGACTCA, with its PWMS equal to only 7.2119.

It is important to keep in mind that the interpretation of PWMS becomes more difficult with the inclusion of pseudocounts. Without the pseudocounts, the rule of thumb concerning PWMS (or log-odds in general when base 2 is used in logarithm) is that a PWMS greater than 10 is taken as significant, i.e., $P(S|\theta_{Yes})$ is about 1000 times greater than $P(S|\theta_{No})$. A PWMS of 0 means the two hypotheses are equally likely. This interpretation is not valid with the addition of pseudocounts. For this reason, whenever possible one should compute PWMS without pseudocounts or use very small pseudocounts so that the interpretation above can still be approximately correct. When α = 0.01, PWMS is 12.3897 for NCF4 and 2.0437 for IGLVIV-59 (Recall that, when α = 0.5, SWMS for IGLVIV-59 is 7.2119 which is difficult to interpret because 7.2119 seems to be substantially larger than 0). We may say that the 13-mer from NCF4 is likely a true translation initiation signal, whereas that from IGLVIV-59 has already decayed into background as a pseudogene typically would. In other words, we can infer that it is a pseudogene by looking at the 13-base fragment at its putative initiation site.

Thus, by scanning along the sequence and computing PWMS for each 13-mer, we can quickly obtain the distribution of putative translation initiation signals along an unknown nucleotide sequence. This is particularly useful when one has already identified the ORFs (open reading frames) by other methods and wants to refine the prediction by identifying the exact location of the translation initiation signals. Figure 5-1 illustrates the result of scanning the 5-end of the NCF4 gene.

The peak score (Figure 5-1) corresponds to the 13-mer with 5 bases flanking the initiation ATG. What is particularly interesting is that the scores are generally smaller than 0. Recall that random combination of nucleotides according to the four p_i values should result in the expected value of PWMS equal to 0. The observation that PWMS scores are generally substantially smaller than 0 implies that, other than the 13-mer with the initiating ATG in the middle, selection has acted in such a way as to make all other 13-mer's to contain weaker translation initiation signals than randomly assembled sequences. In other words, there is selection to reduce conflicting translation initiation signals in the 5'-end of coding genes.

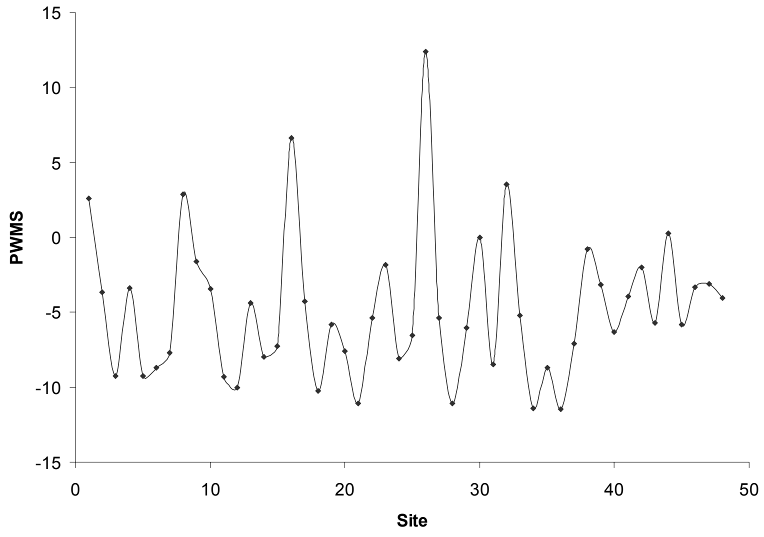


Figure 5-1. Illustration of scanning the 5'-end of the NCF4 gene (30 bases upstream of the initiation codon ATG and 27 bases downstream of ATG. The highest peak, with PWMS = 12.3897, corresponds to the 13-mer with 5 bases flanking the ATG. PWMS computed with α = 0.01.

Note that the interpretation above depends much on the choice of the α in computing pseudocounts in Eq. (5.21). If α is substantially larger than 0, e.g., in the range of 0.1 or larger, then a sequence generated from a random combination of the four p_i values will no longer be expected to have a PWMS of 0, and the above interpretation would be quite inappropriate.

There seems to be some periodicity in PWMS (Figure 5-1). It might be worth the effort of applying the fast Fourier transform to quantify the periodicity.

PWM is implemented in my program DAMBE (Xia, 2001; Xia and Xie, 2001b). One may use it to solve practical problems where PWM is appropriate.

3.2 Perceptron

The perceptron is a binary classifier, being one of the simplest artificial neural networks invented in 1957 at the Cornell Aeronautical Laboratory by Frank Rosenblatt (Rosenblatt, 1958). I will abbreviate “artificial neural networks” as just “neural networks” from now on. Perceptron is also one of the earliest neural networks that became widely known and contributed to the increased research effort on neural networks in the early 1960s. The fancy name must have contributed to its early popularity. Had it been labeled just as “a novel classifier”, “a linear classifier” or, even worse, “a linear binary classifier that cannot handle the XOR problem”, it would probably never see its light of day. Note that these latter labels are in fact far more meaningful than the word “perceptron”. It seems that human beings, through some miraculous mechanisms of evolution, have developed a particular fascination with words suffixed with “-tron”. Whoever can understand and exploit this human fascination seems to be one step closer to success. Animals are known to exploit preexisting fascination of other animals to increase their chance of survival and reproduction, leading to the proposal of the sensory exploitation hypothesis (Arnqvist, 2006; Ryan *et al.*, 1990; Sakaluk, 2000).

The perceptron met its bumpy journey in late 1960s. Because of the theoretical objection against the perceptron concept and the highlight of its limitations by Minsky and Papert (1969), enthusiasm and funding for research in artificial intelligence in general and perceptrons in particular decreased substantially until the field was revived again in the 1980s. Although the single-layer Perceptrons were proven to be incapable of learning the "exclusive or (XOR)" operation (Minsky and Papert, 1969), later extensions of the multi-layer perceptrons are able to handle such non-linear problems (Freund, 1998; Lerner *et al.*, 1995; Rossi and Conan-Guez, 2005). An alternative for solving the XOR problem is to use the support vector machine or SVM (Burges, 1998).

Perceptron has been used in bioinformatics research since 1980s. The identification of translational initiation sites in *E. coli* (Stormo *et al.*, 1982a) is perhaps the first publication of applying the perceptron algorithm in identifying gene features. The algorithm has also been used recently for finding the ATP/GTP-binding motif (Hirst and Sternberg, 1991).

The perceptron is conceptually similar to Fisher’s two-group linear discriminant function analysis (Fisher, 1936) for continuous variables, often referred to as LDA (for **l**inear **d**iscriminant function **a**nalysis). LDA is used when we have N variables, M cases, and a M×N matrix, with the first m_1 cases belonging to one group and the next m_2 (=M-m_1) cases belonging to the other group. The objective is to derive a linear function of the N variables to maximize the difference between the two groups (or, in the machine-learning literature, maximize the signal/noise ratio). I will introduce you to Fisher's LDA in the section dealing with content sensors.

With the perceptron for discriminating between two groups of sequences, we designate one group as the positive group and the other as the negative group. The objective is to obtain a weight matrix that can be used to obtain scores for the unknown sequences and assign these sequences different membership according to the scores.

Here is how perceptron works. Suppose we have two groups of sequences designated as POS (for positive) and NEG (for negative) groups, respectively. The following two groups of sequences represent one of the simplest perceptron problems. We will use this fictitious data set to illustrate the perceptron algorithm:

```
POS1 ACGT
POS2 GCGC

NEG1 AGCT
NEG2 GGCC
```

In practical applications, the sequences will typically be longer than 4 and each group will typically contain a very large number of sequences. The two groups do not necessarily have to contain exactly the same number of sequences, although statistics involved would be simpler when they do. Here we will ignore the statistics part entirely and focus only on the algorithmic aspect of perceptron. Note that we will often refer to sequences in the POS group as POS sequences and those in the NEG group as NEG sequences.

Our task is to find a weighting matrix that can be used to assign a score to each sequence in such a way that sequences in the POS group will have high scores (typically larger than 0) and those in the NEG group will have low scores (typically smaller than 0). The weight matrix and sequence scores are the principal output from perceptron training. From now on, we will use symbols S, PS, and W to designate an input sequence, the perceptron score for the input sequence, and the weight matrix, respectively.

The first step is to initialize W, which is a 4×L matrix for nucleotide sequences and a 20×L matrix for amino acid sequences. W should be initialized with non-zero values. You will see why you should not initialize the weighting matrix with all zero values. For molecular sequences, M is typically initialized with values of one (Table 5-3).

The perceptron algorithm involves the iteration of two steps: (1) taking sequences from the NEG and POS groups sequentially (or randomly when a very large number of sequences are present in each group or when the perceptron cannot converge) and computing PS for each input sequence, and

(2) updating the values in W based on PS. For each sequence S, PS is computed as

$$PS = \sum_{j=1}^{L} W_{S_j,j} \tag{5.25}$$

Table 5-3. The weighting matrix (W) for the fictitious example with two sequences of length 4 in each group, initialized with values of 1. The first row designates sites 1-4.

Base	1	2	3	4
A	1	1	1	1
C	1	1	1	1
G	1	1	1	1
T	1	1	1	1

Take the POS1 sequence in the fictitious example, its PS based on the initialized W in Table 5-3 is

$$PS_{POS1} = W_{A,1} + W_{C,2} + W_{G,3} + W_{T,4} = 4 \tag{5.26}$$

In fact, PS will be 4 for every sequence with the freshly initialized W with values of 1. What might be slightly confusing is the updating of W. It is done according to the following rules (there could be slight variation of the rules in different applications):

$$\begin{aligned} &W_{S_j,j} = W_{S_j,j} + 1, \text{ if S is from POS group and PS} < 0 \\ &W_{S_j,j} = W_{S_j,j} - 1, \text{ if S is from NEG group and PS } \geq 0 \\ &\text{No change in } W_{S_j,j} \text{ otherwise.} \end{aligned} \tag{5.27}$$

where j = 1, 2, …, L where L is sequence length (=4 in our fictitious example).

Let us start with the NEG sequences in the fictitious example. Note that you would waste computational time by starting with sequences in the POS group because the resulting PS will all be 4 and consequently, according to the rules for updating W, no updating is made with positive PS from POS sequences when PS = 4 > 0.

PS for NEG1, i.e., S = AGCT is 4. According to the rules for updating W in Eq. (5.27), we should update W by reducing the relevant W_{ij} values by 1. The updated W is shown in Table 5-4a, with $W_{A,1}$, $W_{G,2}$, $W_{C,3}$ and $W_{T,4}$ in the original W (Table 5-3) reduced by 1, with updated values highlighted in bold.

The next input sequence is NEG2 (=GGCC) which has PS = 2 based on the updated W in Table 5-4a. According to the rules of updating W in Eq. (5.27), we should again subtract 1 from W_{ij} values corresponding to the sequences, i.e., $W_{G,1}$, $W_{G,2}$, $W_{C,3}$ and $W_{C,4}$ all have their values reduced by 1. This update changes the weighting matrix from that in Table 5-4a to that in Table 5-4b.

Table 5-4. The first round of the training process in the perceptron algorithm. Updated values are highlighted in bold.

NEG1: AGCT, PS = 4, update	Base	1	2	3	4
	A	**0**	1	1	1
(a)	C	1	1	**0**	1
	G	1	**0**	1	1
	T	1	1	1	**0**
NEG2: GGCC, PS = 2, update	A	0	1	1	1
	C	1	1	**-1**	**0**
(b)	G	**0**	**-1**	1	1
	T	1	1	1	0
POS1: ACGT, PS = 2, no update	A	0	1	1	1
	C	1	1	-1	0
(c)	G	0	-1	1	1
	T	1	1	1	0
POS2: GCGC, PS = 2, no update	A	0	1	1	1
	C	1	1	-1	0
(d)	G	0	-1	1	1
	T	1	1	1	0

We can proceed with POS1 and POS2 sequences, but both have PS = 2 and, according to the rules of updating W, no change should be made. At this point, no input sequence will lead to updating of W, i.e., the two NEG sequences will both have PS = -2 and the two POS sequences will both have PS = 2. So we conclude that the perceptron has already converged.

One problem with the perceptron algorithm is that it may not converge, e.g., when it is applied to solving an XOR problem that will be detailed latter. For this reason, computer programs implementing the perceptron algorithm will allow you to input a maximum number of iterations.

You might have noticed that some cells may never be involved in computing PS for any input sequence, especially when the training set contains few sequences. This could cause problems in using W for classifying unknown sequences. For example, a sequence of TAAA would have a score of 4 and we would consequently assign it to the POS group although it bears no similarity to any sequences in the training sequences in the POS group. The reason for this is that the W_{ij} values corresponding to

TAAA are never involved in computing PS during the training process and have still retained the initial value of 1.

To avoid this problem, we will take the final W from perceptron training in Table 5-4 and set to 0 all W_{ij} values not involved in computing PS for any input sequences. The post-processed W is shown in Table 5-5. An alternative is to set to 0 all W_{ij} values that have never been updated during the iteration.

Table 5-5. Final W after setting all W_{ij} values not involved in computing PS to 0. Those involved in computing PS are highlighted in bold.

Base	1	2	3	4
A	**0**	0	0	0
C	0	**1**	**-1**	**0**
G	**0**	**-1**	**1**	0
T	0	0	0	**0**

With the postprocessed W, the two POS sequences will still have PS = 2 and the two NEG sequences still have PS = -2. However, for a sequence such as TAAA, PS = 0, i.e., the perceptron is absolutely unable to classify it into either the POS or the NEG group. The final W (Table 5-5) , trained with the extremely limited training set of only two sequences in the POS and the NEG groups, shows explicitly that it can only classify sequences with C or G in the second and third sites because all values in the first and last columns are zero (Table 5-5). You may find this obvious by looking at the four short training sequences.

Some readers have criticized me for not using a “biologically more realistic” example instead of the 4-nucleotide sequences. My objective is equivalent to demonstrating a rainbow with just a few droplets of water. Those who insist on seeing a real rainbow spanning the sky would probably have to find one for themselves in their practical research.

The perceptron is ideally suited for two groups of objects (be they sequences or any other objects) that are linearly separable. This can be better understood in comparison with what are not linearly separable (e.g., the XOR problem). So we will illustrate both in an intuitive way, with nucleotide sequences in the form of “ACGTXYACGT”. Suppose that the “XY” is “TT” in sequences of the POS group, and can be any other non-“TT” dinucleotides in the NEG group. These two groups of sequences can be easily separated by the perceptron algorithm. Note that the XY in the sequences contains either 0, 1 or 2 T’s, and the sequences in the positive group, with their XY containing 2 T’s, is linearly separable from sequences with their XY containing 0 or 1 T’s.

Now suppose the XY in our sequences in the POS group contains only one T, i.e., either X or Y is a T but not both. The XY in the sequences in the negative group can either be TT or contain no T (e.g., XY = “CG”). This is one of the simplest XOR problems that a conventional perceptron cannot handle. The iteration will continue forever without convergence. Note that XY in POS sequences in this case contains only one T and they are no longer linearly separable from the NEG sequences containing either 0 or two T's. This XOR problem has plagued the first application of the perceptron algorithm to the study of the translation initiation sites in *Escherichia coli* (Stormo *et al.*, 1982a), resulting in failure to converge on a solution to separate the sequences with the translation initiation site from those without.

The perceptron has not been used often to solve problems in molecular biology and bioinformatics, and this is mainly caused by the overemphasis that perceptrons cannot deal with the XOR problem. In fact, many XOR problems can be reduced to non-XOR problems and be solved easily by perceptrons. For example, the XOR problem in the previous paragraph can be reduced to a non-XOR problem by using dinucleotide sequences, i.e., AC, CG, GT, TX, XY, ..., GT and then solved by the perceptron method. In this case, we will have a W with dimensions 16×(L-1) because there are 16 possible dinucleotides.

In the previous section on position weight matrix, we have used a set of 508 named protein-coding genes and, from each gene, extracted the 13-mer for illustrating the computation involved. We may use that set of 13-mers as the POS group and randomly generate 1000 (or more) 13-mers with the same nucleotide frequencies. These two set of sequences can then be fed into perceptron for analysis. Table 5-6 shows the output of the weight matrix from the dinucleotide-encoded sequences.

Table 5-6. Weighting matrix obtained with the dinucleotide-encoded perceptron. Frequent dinucleotides at sites flanking the ATG codon are in bold. The last row list either the most frequent dinucleotide at the site or, if no dinucleotide is more frequent than others, the consensus dinucleotides.

	1	2	3	4	5	6	7	8	9	10	11	12
AA	-1	-2	0	-1	0	-2	-1	-1	0	-1	-1	0
AC	-1	-3	**1**	2	-2	-2	-2	-2	-2	1	-4	-1
AG	1	-1	-4	-1	-2	-2	-3	-4	-2	-2	-2	-3
AT	1	-2	-1	0	-1	**9**	-1	1	-1	-1	1	1
CA	-2	**4**	0	1	**1**	-2	0	-2	-1	0	0	0
CC	1	-2	-5	-1	-2	-1	-1	-2	0	-3	-1	0
CG	-2	0	-1	-3	-3	-1	-4	-3	-5	0	-2	-6
CT	-1	-3	-2	-2	-1	0	-1	1	-1	1	0	2
GA	-2	-1	**1**	-2	**1**	-1	-2	0	-2	-1	-1	-2
GC	-3	-2	**1**	-7	1	-3	-2	-1	1	0	0	0
GG	-3	-1	-4	-1	-4	-2	-2	**3**	0	-3	-2	-4
GT	-3	1	**1**	0	-5	-5	0	-1	-5	-2	-4	0
TA	-3	-5	-1	-3	**1**	-2	-3	-2	0	-4	-1	-1
TC	**2**	-1	-2	1	-1	0	-1	-2	-1	-1	1	-3
TG	1	1	-1	-2	-2	-1	**10**	-2	1	-2	0	0
TT	-3	-1	-1	1	1	-3	-5	-1	0	0	-2	-1
	TC	CA	RN	NN	NA	AT	TG	GG				

The weight matrix in Table 5-6 reached convergence after only 19 iterations. The final weight matrix (Table 5-6) can be used to assign scores to 13-mers, with those having high scores more likely to contain the translation initiation signals. The most frequently or consensus dinucleotides in Table 5-6 can be used to reconstruct a consensus sequence (Figure 5-2). In this particular case, the consensus happens to contain the Kozak translation initiation consensus.

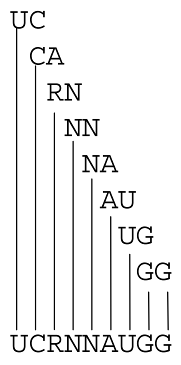


Figure 5-2. Construction of a consensus (UCRNNAUGG) from overlapping dinucleotides.

To summarize, the input to a perceptron consists of two groups of sequences of the same length and our objective is to find a scoring function to maximize the difference between the two groups. The scoring function is in the form of a weighting matrix, derived from training the perceptron with the two groups of sequences. The weighting matrix can be used to compute a score for a sequence according to Eq. (5.25). The output from training a perceptron is a weighting matrix and a score for each input sequence. A perceptron that has achieved convergence will have positive sequence scores in the POS group and negative sequence scores in the NEG group. An unknown sequence is classified into the POS group when its score is larger than 0, and into the NEG group when its score is smaller than 0. Perceptron algorithms are implemented in my program DAMBE (Xia, 2001; Xia and Xie, 2001b).

In one study using the perceptron algorithm to characterize the translation start site (TSS) in *E. coli* (Stormo *et al.*, 1982b) involving 124 true TSSs and 78000 other sites (OSs), the perceptron was unable to converge. However, all 124 true TSSs have perceptron score (PS) greater than 0, whereas only 64 out of 78000 OSs have PS greater than 0. Now if we have a site with PS > 0, what is the probability that it is a true TSS?

A student with a Bayesian inclination may go through a circuitous route to answer the question as follows. Define Y as the event of having a site with

PS > 0. Let θ_{Yes} stand for the hypothesis of the site being a true TSS, and θ_{No} the hypothesis of the site not being a TSS. The two prior probabilities are

$$\begin{aligned} P(\theta_{Yes}) &= 124/78124 = 0.001587 \\ P(\theta_{No}) &= 1 - P(\theta_{Yes}) = 0.998413 \end{aligned} \tag{5.28}$$

The two likelihood functions are

$$\begin{aligned} P(Y \mid \theta_{Yes}) &= 124/124 = 1 \\ P(Y \mid \theta_{No}) &= 64/78000 = 0.000821 \end{aligned} \tag{5.29}$$

Now the answer to the question (i.e., what is the probability of a site being a true TSS given its PS > 0) is

$$\begin{aligned} P(\theta_{Yes} \mid Y) &= \frac{P(Y \mid \theta_{Yes})P(\theta_{Yes})}{P(Y \mid \theta_{Yes})P(\theta_{Yes}) + P(Y \mid \theta_{No})P(\theta_{No})} \\ &= \frac{1 \times 0.001587}{1 \times 0.001587 + 0.000821 \times 0.998413} = 0.659 \end{aligned} \tag{5.30}$$

This calculation is correct but unnecessary. A smart student will note that there are a total of 188 (= 124 + 64) sites that have PS > 0 (i.e., event Y is observed). Out of these 124 are true TSSs (i.e., θ_{Yes} is true). Thus, by definition

$$P(\theta_{Yes} \mid Y) = 124/188 = 0.6596 \tag{5.31}$$

4. CHARACTERIZING FEATURES OF CONTENT SENSORS

Content sensors are typically frequencies of various kinds, e.g., nucleotide, dinucleotide, and trinucleotide frequencies. For example, triplet frequencies or nucleotide frequencies at the three different positions of the triplets are often different between exons and introns. Once a sequence is compiled into these frequency tables, the site-specific information is lost. Any sensor that does not include site-specific information is a content sensor and is the topic in this section.

We will illustrate a few special content sensors that can help discriminate between coding exons and non-coding sequences (e.g., introns), based on the sequence pattern created by DNA methylation. DNA methylation is a process that simply cannot be avoided in a book with the word "cell" in its title. In short, DNA methylation plays a key role in gene regulation in many vertebrate species, and is a ubiquitous biochemical process particularly pronounced in vertebrate genomes, with its main function being tissue-specific gene regulation (Bestor and Coxon, 1993; Rideout *et al.*, 1990; Sved and Bird, 1990). A typical representative of the vertebrate methyltransferase is the mammalian DNMT1 with five domains of which the NlsD, ZnD and CatD domains bind specifically to unmethylated CpG, methylated CpG and hemimethylated CpG sites, respectively (Fatemi *et al.*, 2001). Methylation of C (at its #5 carbon atom) in the CpG dinucleotide greatly elevates the mutation rate of C to T through spontaneous deamination of the resultant m^5C (Brauch *et al.*, 2000; Tomatsu *et al.*, 2004), where the superscript 5 indicates the #5 carbon atom, generating strong footprints in both prokaryotic and vertebrate genomes (Xia, 2003, 2004, 2005a).

4.1 Indices of content sensors related to DNA methylation and spontaneous deamination

Here we develop indices to capture the differential substitution patterns in coding and non-coding sequences. Designate the nucleotide frequencies of a sequence as p_A, p_C, p_G and p_T, and the sequence length as L. Consider both coding and non-coding sequences as a linear sequence of consecutive non-overlapping triplets, and the nucleotide frequencies at the three sites of a triplet as p_{i1}, p_{i2} and p_{i3}, where i = 1, 2, 3, and 4 corresponding to nucleotide A, C, G, and T, respectively.

For non-coding sequences, there is no codon structure. So we expect $p_{i1} \approx p_{i2} \approx p_{i3} \approx p_i$, where p_i is the average of p_{i1}, p_{i2}, and p_{i3}. For coding sequences, various mutation and selection processes will create heterogeneity in nucleotide frequencies among the three sites (Xia, 1998a). Take NCG codon for example, where N stands for any of the four nucleotides. DNA methylation and spontaneous deamination tend to change these NCG codons to NTG and NCA codons (the latter resulting from CpG$\rightarrow$TpG mutations in the complementary strand), with the former change being nonsynonymous and the latter synonymous. Because nonsynonymous substitution is generally deleterious and consequently selected against by natural selection, they are much rarer than the synonymous substitutions in a large number of protein-coding genes in many organisms studied (Xia, 1998a; Xia *et al.*, 1996; Xia and Li, 1998), we should expect NCG$\rightarrow$NCA mutations more often than NCG$\rightarrow$NTG mutations. This tends to increase the frequency of A at the third codon position. Similarly, dicodons such as "NNC GNN" tend to mutate synonymously to "NNT GNN" with DNA methylation and spontaneous deamination, increasing the frequency of T at the third codon position. Thus, in contrast to non-coding sequences where we expect $p_{i1} \approx p_{i2} \approx p_{i3} \approx p_i$, we should expect $p_{i1} \neq p_{i2} \neq p_{i3} \neq p_i$ in coding sequences. This suggests that the deviation of p_{ij} (where j = 1, 2 and 3 corresponding to the three triplet sites) from p_i can contribute to the discrimination between coding and non-coding sequences. A measure of this deviation that is independent of L is as follows:

$$\varphi_{Nuc} = \frac{\sum_{i=1}^{4} \sqrt{\frac{\sum_{j=1}^{3} \frac{(f_{ij} - f_i)^2}{f_i}}{N_i}}}{4} = \frac{\sum_{i=1}^{4} \left(\frac{f_{i1}^2 + f_{i2}^2 + f_{i3}^2}{3 f_i^2} - 1 \right)}{4} \tag{5.32}$$

where f_{ij} stands for the number of nucleotide i at triplet position j, f_i is the mean number of nucleotide i averaged over the three codon (triplet) positions, and N_i is the sum of nucleotide i in the sequence. We expect φ_{Nuc} to be greater for coding sequences than for non-coding sequences.

Following a similar line of reasoning, we expect the dinucleotide frequencies at triplet positions (1,2), (2,3) and (3,1) to be similar to each other in non-coding sequences but different in coding sequences. Designate the number of dinucleotides as $f_{ij.k}$, where ij = AA, AC, ..., TT, respectively, and k = 1, 2, 3 corresponding to the triplet positions (1,2), (2,3) and (3,1), respectively. The deviation of $f_{ij.k}$ from f_{ij}, which is the number of dinucleotide i averaged over the three triplet positions, should also contribute to the discrimination between coding and non-coding sequences. A measure of this deviation that is independent of L is:

$$\varphi_{DiNuc} = \frac{\sum_{i=1}^{4} \sum_{j=1}^{4} \left(\frac{f_{ij1}^2 + f_{ij2}^2 + f_{ij3}^2}{3 f_{ij}^2} - 1 \right)}{16} \tag{5.33}$$

For short sequences, f_{ij} may be zero, in which case φ_{DiNuc} is not defined, or very small, in which case φ_{DiNuc} would fluctuate widely. To avoid this problem, the computation can be done by setting valid f_{ij} as a value larger than a certain number, e.g., 6, and the denominator will be the number of valid f_{ij} values instead of a fixed 16.

DNA methylation and spontaneous deamination decrease the CG-containing triplets and increase the UG- and CA-containing triplets. However, their effect is stronger on introns than on coding sequences because of weaker selection constraints on introns than on coding sequences, e.g., all CGN→TGN, CGN→CAN and NCG→NTG mutations are nonsynonymous. Nonsynonymous mutations are generally deleterious (Xia and Li, 1998) and tend to be selected against in coding sequences but not in non-coding sequences. For this reason, the intensity of methylation effect (designated I_m) should be greater in introns than in coding sequences:

$$I_m = \frac{(f_{NUG,UGN,NCA,CAN} - f'_{NUG,UGN,NCA,CAN}) - (f_{NCG,CGN} - f'_{NCG,CGN})}{f_{NUG,UGN,NCA,CAN} + f_{NCG,CGN}} \qquad (5.34)$$

where f is the sum of frequencies of those subscripted codons, and f' is the corresponding expectation computed simply by

$$f'_{ijk} = N_{triplet} P_i P_j P_k \qquad (5.35)$$

where $N_{triplet}$ is the total number of non-overlapping triplets in the sequence. A more reasonable expectation would be (by taking AAA and AAG for illustration):

$$\begin{aligned} f'_{AAA} &= N_{Lys} \bullet \frac{f_{AAA}}{f_{AAA} + f_{AAG}} \\ f'_{AAG} &= N_{Lys} \bullet \frac{f_{AAG}}{f_{AAA} + f_{AAG}} \end{aligned} \qquad (5.36)$$

where N_{Lys} is the number of triplets identical to lysine codons. However, such a formulation is not equally applicable to non-coding sequences. Note that the reading frame is usually unknown. So one will need to compute over six possible reading frames (three on each strand).

Among UG- and CA-containing codons that tend to be increased by DNA methylation of CpG dinucleotides, five (AUG, CAA, CAC, CAU, and UUG) are generally avoided in coding sequences in vertebrate genomes, either caused by reduced amino acid usage or other unknown factors. Designating these avoided UG- and CA-containing triplets as f_1 and the other UG- and CA-containing triplets as f_2, we define the triplet avoidance index as

$$I_{ta} = \frac{(f_2 - f_2^{'}) - (f_1 - f_1^{'})}{f_1 + f_2} \tag{5.37}$$

Polypurine and polypyrimidine stretches are ubiquitous among eukaryotic genomes (Birnboim *et al.*, 1979; Mills *et al.*, 2002; Ohno *et al.*, 2002), but their frequencies in coding sequences are constrained by the necessity of codons with mixed purines and pyrimidines. For this reason, the polypurine and polypyrimidine triplets tend to be more frequent in non-coding sequences than in coding sequences. We define the following index to measure the tendency of polypurine and polypyrimidine triplets:

$$I_{pp} = \frac{(f_{RRR,YYY} - f_{RRR,YYY}^{'}) - (f_{Mixed} - f_{Mixed}^{'})}{f_{RRR,YYY} + f_{Mixed}} \tag{5.38}$$

4.2 Are these indices useful in discriminating between coding and non-coding sequences?

Here we demonstrate the utility of these indices in discriminating between introns and coding sequences by using the annotated DNA sequence of human chromosome 22 (ref_chr22.gbk) in GenBank. Human chromosome 22 is perhaps the best annotated human chromosome sequence being the first to be sequenced and having undergone many revisions. The CDSs, exons and introns were extracted according to the sequence annotation in the FEATURES table, and their triplet/codon frequencies were computed, by using DAMBE (Xia, 2001; Xia and Xie, 2001b). The indices shown in Eq. (5.32)-(5.38) were also computed by DAMBE for introns and CDSs. Whether the indices are useful can be assessed by how well they can discriminate between coding sequences and introns.

We will take a first look at the coding and non-coding sequences that are at least 2000 bases long. As I have mentioned before, the indices are likely to fluctuate widely with short sequences. Focusing on long sequences will allow us to extract the difference in these indices between the two groups more efficiently. We will refer to these sequences with $L \geq 2000$ as the training set and will use the discriminant function derived form this training set to screen other human chromosome 22 sequences.

The mean and standard deviation of the five indices (Table 5-7) reveal substantial difference in these indices between the coding and non-coding (intron) sequences, in the direction as we have expected. For example, the mean φ_{Nuc} and φ_{DiNuc} are both much smaller in introns than in coding sequences (Table 5-7), with p = 0.0000. We will subject this training set to Fisher's two-group linear discriminant analysis (Fisher, 1936) to obtain a function that can be used to assign an unknown sequence to either the coding or non-coding group.

Table 5-7. Mean and standard deviation of the five indices defined in Eqs. (5.32)-(5.38) for coding and non-coding sequences in the training set. For intron sequences, the indices differ little for the six different triplet frames (i.e., 3 on each strand), and the numerical results are presented only for the triplet frame starting with the first intron site.

SeqName	SeqType	φ_{Nuc}	φ_{DiNuc}	I_{pp}	I_m	I_{ta}
CELSR1	Coding	0.3508	0.5257	-0.0734	0.3073	-0.1205
MYO18B	Coding	0.2329	0.3169	0.2074	0.4091	0.0298
EP300	Coding	0.1865	0.2780	0.1519	0.4748	0.0747
PKDREJ	Coding	0.1525	0.2146	0.1185	0.2551	0.0130
CACNA1I	Coding	0.3670	0.5595	-0.0173	0.3484	-0.1476
…	…	…	…	…	…	…
Mean		0.2507	0.3822	0.1069	0.3790	-0.0219
Std		0.0775	0.1090	0.0854	0.1029	0.1720
LOC40205	Intron	0.0046	0.0110	0.2194	0.4284	0.2308
SYN3	Intron	0.0033	0.0081	0.2318	0.4266	0.2100
LARGE	Intron	0.0107	0.0178	0.2131	0.3983	0.2376
OSBP2	Intron	0.0046	0.0124	0.2050	0.4382	0.2394
SEZ6L	Intron	0.0034	0.0126	0.2307	0.4216	0.2363
…	…	…	…	…	…	…
Mean		0.0318	0.0768	0.2106	0.4348	0.2362
Std		0.0166	0.0302	0.0499	0.0825	0.0821

I have previously mentioned the similarity between perceptron and Fisher's discriminant analysis. The former takes two groups of vectors (which is not limited to nucleotide or amino acid sequences) and produce a weighting matrix that can be used to obtain sequence scores to allow the assignment of unknown sequences to either one or the other group. The latter takes a data matrix made of two groups as in Table 5-7 and generate a linear discriminant function which is also used to generate a score to allow the assignment of unknown sequences to either one or the other group.

The computation involved in the two-group discriminant function analysis is not difficult and the method is implemented in my program DAMBE (Xia, 2001; Xia and Xie, 2001b). Running the analysis on the data generates the desired discriminant function and its associated statistics (Table 5-8).

Table 5-8. Coefficients of the two-group discriminant function (DiscFunc).

	φ_{Nuc}	φ_{DiNuc}	I_{pp}	I_m	I_{ta}
Mean	0.1413	0.2295	0.1587	0.4069	0.1072
DiscFunc	69.7357	-3.9504	-0.2009	-0.0802	-4.2111

Just as the weighting matrix from training a perceptron can be used to obtain a score for an unknown sequence for assigning it to either one group or the other, the coefficients of the discriminant function in Table 5-8 can be used to get a score for a sequence with its five computed indices for classifying it to either one group or the other. The score for each sequence is computed as

$$S = a_1(\varphi_{Nuc} - \overline{\varphi}_{Nuc}) - a_2(\varphi_{DiNuc} - \overline{\varphi}_{DiNuc}) - a_3(I_{pp} - \overline{I}_{pp}) \\ - a_4(I_m - \overline{I}_m) - a_5(I_{ta} - \overline{I}_{ta}) \qquad (5.39)$$

where a_1 to a_5 are in the second row of Table 5-8, and the five means are in the first row of Table 5-8. For example, if we have a sequence whose five indices are computed to be 0.0566, 0.1969, 0.1201, 0.7027, and 0.5351, respectively, its S value according to Eq. (5.39) will be -4.6763. As it is smaller than 0, it is assigned to the second (i.e., intron) group. A sequence with its five indices equal to 0.4098, 0.6509, 0.0571, 0.3916, and -0.0553, respectively, would have a S value equal to 20.6884, resulting its assignment to the first (i.e., coding sequence) group. Thus, by computing the indices based on sequence information only, we can predict whether it is coding or non-coding.

The classification of the training set of 1935 sequences ($L \geq 2000$) by the discriminant function is quite promising (Table 5-9), with only 8 misclassifications out of 1935 sequences. It is prone for one to claim an error rate of 0.0041 (= 8/1935), which is both misleading and meaningless. For example, if we have two groups with 2 coding and 2000 non-coding sequences, even a blind classifier that assigns every sequence as non-coding would have only 2 misclassifications out of the 2002 sequences. Should you claim that this is a very good classifier with an error rate of only 0.001 (=2/2002)?

A more acceptable error assessment is by the weighted error rate expressed as follows

$$\varepsilon_w = \frac{\dfrac{N_{1.wrong}}{N_1} + \dfrac{N_{2.wrong}}{N_2}}{2} \qquad (5.40)$$

where N_1 and N_2 are the number of sequences in the first and second groups, respectively, and $N_{1.wrong}$ and $N_{2.wrong}$ are the misclassified cases in the first and second groups, respectively. This error rate for the training set is 0.0276 (Table 5-9). For the fictitious example with 2 coding and 2000 non-coding sequences and a blind classifier that assigns all sequences to the non-coding group, $\varepsilon_w \approx 0.5$, which reveals the truth that the blind classifier is no better than a random classifier.

Table 5-9. Results of classification with the discriminant function.

		Classified to		
L (bases)	From	CDS	Intron	Error
≥2000	CDS	105	6	0.0276
	Intron	2	1822	
1000-1999	CDS	225	18	0.0376
	Intron	1	876	
500-999	CDS	155	23	0.0717
	Intron	10	696	
200-499	CDS	80	29	0.2494
	Intron	156	514	

I wish to illustrate the discriminating power of these indices by a particular “intron” that has its index values similar to coding sequences, and is classified by the linear discriminant function (Table 5-9) as a coding sequence. The “intron” belongs to a gene annotated as “LOC284861” in the ref.chr22.gbk file, starts with GT and ends with AG, and is “derived by automated computational analysis” according to the FEATURES table in the GenBank file. However, it is annotated as part of the coding sequence in other cloned homologous human cDNA sequences (GenBank accession: AL117481, AL122069, AL133561).

There are three lines of evidence to suggest that this “intron” is not an intron. First, when the intron and its two flanking exons are treated as a single exon, there is no embedded stop codon. Second, it has at least four indels when aligned with the GenBank sequence XM_375042, and all indels are inframe triplets. Such indel events are typical of coding sequences. Third and perhaps the most important, aligning the “intron” with other homologous human cDNA genes shows that its starting GT and ending AG are not conserved, which is not what we would expect if the starting GT and ending AG represent true donor and acceptor sites. All these suggest that the “misclassification” of the intron as a coding sequence by the discriminant function may in fact represent correct identification.

Another indication of the discrimination power of these indices is that when sequences annotated as hypothetical genes are excluded, then the error rate of the classification is decreased overall by nearly one order of magnitude. This suggests that a high proportion of hypothetical genes are not true genes.

The discriminant function (Table 5-8) derived from this training set can be used successfully in discriminating between the CDS and the intron sequences not in the training set, but the power of discrimination offered by these five indices decreases with decreasing sequence length (Table 5-9). Many exons in eukaryotic genomes are quite short, highlighting the difficulty in gene prediction.

At this point I would like to introduce you to a classroom example typically used to illustrate the hidden Markov models. The example involves a dishonest casino dealer who switches between a fair die and a loaded die (i.e., two hidden states). If the loaded die differ much from the fair die, e.g., if the probability of having 6 is nearly 1 for the loaded die, then a short stretch of 6's is sufficient to identify the point of switching. Note that a stretch of 6's implies a high frequency of 6 which is equivalent to a content sensor. However, if the casino dealer switches to the loaded die, tosses it only once, and immediately switches back to the fair die, then it becomes very difficult to catch her (I am not sure if I should use him or her but most new books seem to use "her" as default.). The same applies to gene prediction. If exons and introns (hidden states) are long, then it is easy to identify them. If they are short, then it is often theoretically impossible to identify them by content sensors only.

The next chapter is on hidden Markov models, which uses information in both signal and content sensors. It is a powerful computational tool that anyone interested in bioinformatics should not miss.

Chapter 6

HIDDEN MARKOV MODELS

1. INTRODUCTION

In the previous chapter we have developed basic understanding of signal and content sensors in structure-based gene prediction. Hidden Markov models (HMMs) can incorporate information in both signal and content sensors, and is frequently applied in structure-based gene prediction (Baldi and Brunak, 2001; Durbin, 1998; Pevzner, 2000). However, HMMs are also used in many other contexts. For this reason I have intentionally chosen to illustrate the application of HMMs in predicting protein secondary structure instead of in gene prediction. However, I do expect you to know how to use HMMs in gene prediction once you have learned how to use HMMs in predicting protein secondary structure.

Before illustrating the utility of HMMs, we will first give a brief introduction to Markov models sufficient for comprehending HMMs. In particular, we wish to know (1) two categories of parameters, the frequency parameters and the rate parameters in the transition probability matrix that characterize a 1st-order Markov model, also known as a Markov chain, (2) how to obtain the equilibrium frequencies, and (3) how to calculate the likelihood of a given sequence of events that follow a Markov model. These are the minimal requirement for a reasonable understanding of HMMs.

HMMs are illustrated numerically because most readers, just like me, cannot see the beauty of equations until they are rendered to numbers. You will learn the essential elements in a HMM, how to train a HMM, how to reconstruct the most probable path of hidden states by using the Viterbi algorithm, how to compute the likelihood of a particular sequence of events by using the forward algorithm, and how to estimate parameters in a HMM. If you are a programmer, you will be able to implement the algorithms associated with HMM once you have finished this chapter.

2. MARKOV MODELS

A Markov model is typically used to model the dynamic change of a random variable over time. For example, the pitch of music from your stereo reaching your ear over time t_1, t_2, …, t_k can be represented as X_{t1}, X_{t2}, …, X_{tk}. The X_{ti} values constitute a realization of the random variable X. However, Markov models are equally applicable over a one-dimensional space. For example, along the linear DNA, nucleotides change over site 1, 2, …, i and can be represented as X_1, X_2, …, X_i.

Suppose a DNA sequence of length L, with each site coded only as purine (R) and pyrimidine (Y). The proportions of R and Y are designated as p_R and p_Y, respectively. What is the probability that a nucleotide at site i, designated as X_i, is R (or Y)? Without any further information, our prediction of a site being occupied by R (or Y) is simply p_R (or p_Y). For a fictitious sequence of alternating purine and pyrimidine triplets,

S = RRRYYYRRRYYY……,

$p_R = p_Y = 0.5$. For the partially sequenced human chromosome 22 with 10 contigs (accession NT_028395, NT_011519, NT_011520, NT_011521, NT_011523, NT_011525, NT_019197, NT_113818, NT_011526, NT_113961, dated 02-MAR-2006) with 35,017,877 base pairs (bp), p_R and p_Y are nearly equal, being 0.50047 and 0.49953, respectively. Without any further information, our prediction of a site being occupied by R and Y in human chromosome 22 is 0.50047 and 0.49953, respectively. These are also our best possible estimates of p_R and p_Y when there is no site dependence.

Now we consider a slightly more realistic case with the minimal site dependence, with the probability of X_{i+1} being either R or Y depending only on whether X_i is R or Y. We use P_{RR}, and P_{RY} to designate the conditional probabilities of R and Y, respectively, at site i+1 given R at site i, and P_{YR} and P_{YY} to designate the conditional probabilities of R and Y at site i+1 given Y at site i. Some authors (Higgs and Attwood, 2004, p. 234; Weir, 1990, p. 238) give the estimate of the conditional probabilities in the following form:

$$P_{RR} = \frac{N_{RR}}{N_R} \tag{6.1}$$

where N_R and N_{RR} are the number of R's and RR doublets in the sequence. What they have in mind is a very long sequence. Suppose we have only a short sequence S' equal to the first 12 nucleotides of S, i.e.,

S' = RRRYYYRRRYYY.

Now Eq. (6.1) will result in weird estimates. For example, P_{YY} and P_{YR} would be 2/3 and 1/6, respectively. These P_{YY} and P_{YR} values have two problems, one major and one minor. The major one is that they are probabilistically incorrect because the two do not even sum up to 1 as they should. The minor one is that, given the regular alternating patterns of purine triplets and pyrimidine triplets in S', our intuition suggests that P_{YR} should be equal to P_{RY}. The estimated P_{YR} (= 1/6) being half of P_{RY} (= 2/6) according to Eq. (6.1) makes us feel uncomfortable. A mathematically more consistent alternative for P_{YY} and P_{YR} (which is also a maximum likelihood estimate) is

$$P_{XK} = \frac{N_{XK}}{\sum_{K'} N_{XK'}} \tag{6.2}$$

where the denominator is the summation of all dinucleotides starting with nucleotide X. This yields P_{YY} = 4/5 and P_{YR} = 1/5. The two now do sum up correctly to 1, eliminating the major problem we mentioned above. Moreover, the difference between P_{YR} and P_{RY} is now somewhat smaller, alleviating the minor problem. A still more reasonable estimate of P_{XK}, assuming that the last nucleotide is followed by R with a probability p_R and by Y with a probability p_R, is

$$P_{YR} = \frac{N_{YR} + p_R}{1 + \sum_{K'} N_{YK'}};\ P_{YY} = \frac{N_{YY} + p_Y}{1 + \sum_{K'} N_{YK'}} \tag{6.3}$$

For sequence S, P_{RR} = 2/3, P_{RY} = 1/3, P_{YR} = 1.5/6 and P_{YY} = 4.5/6 according to Eq. (6.3). Thus, the major problem is again solved and the minor problem is further alleviated although P_{YR} and P_{RY} are still not the same as from our intuition.

To simplify the presentation of Markov models, let us just assume that we have a long sequence of alternative purine triplets and pyrimidine triplets, so that indeed $P_{RR} = P_{YY}$ = 2/3 and $P_{YR} = P_{RY}$ = 1/3. These four values, arranged in a 2×2 matrix (Table 6-1), constitute what is called the transition probability matrix for a 1st-order Markov model which is also known as a Markov chain. The four corresponding elements for human chromosome 22 are also included in Table 6-1. We note that a purine is more likely to be followed by a purine than by a pyrimidine and that a pyrimidine is more likely followed by a pyrimidine than by a purine, and that this pattern, obvious enough for S, is also true for human chromosome 22 (Table 6-1). This helps us to predict the probability of X_{i+1} given X_i. Take S for example, without information on X_i, our prediction of X_{i+1} being R is p_R (= 0.5). In contrast, with X_i = R, our prediction of X_{i+1} being R is P_{RR} (= 2/3).

Table 6-1. Two transition probability matrices, one for S and one for human chromosome 22, for the Markov chain with two states (R and Y). Note that values in each row in a transition probability matrix are constrained with a row sum of 1.

	S		Human Chr22	
	R	Y	R	Y
R	2/3	1/3	0.56683	0.43317
Y	1/3	2/3	0.43398	0.56602

Other than the transition probability matrix illustrated in Table 6-1, we need to know the frequency parameters, typically arranged in the form of a vector designated by p_i, to characterize the Markov chain. For our example with only two states (R and Y), the two frequency parameters are the probabilities of R and Y at site i, designated by $p_{R.i}$ and $p_{Y.i}$, respectively. In other words, the vector p_i contains two elements, $p_{R.i}$ and $p_{Y.i}$.

We now learn how to obtain equilibrium frequencies of R and Y given the transition probability matrix. Suppose X_{i-1} = R, so $p_{R.i-1} = 1$ and $p_{Y.i-1} = 0$, and $p_{R.i}$ and $p_{Y.i}$ are naturally equal to P_{RR} and P_{RY}, respectively. In matrix algebra, and using the transition probability matrix for sequence S (Table 6-1), we have

$$\begin{aligned} p_i = p_{i-1}M &= \begin{bmatrix} p_{R.i-1} & p_{Y.i-1} \end{bmatrix} \begin{bmatrix} P_{RR} & P_{RY} \\ P_{YR} & P_{YY} \end{bmatrix} \\ &= \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} 2/3 & 1/3 \\ 1/3 & 2/3 \end{bmatrix} = \begin{bmatrix} 2/3 & 1/3 \end{bmatrix} \end{aligned} \tag{6.4}$$

At site i+1, i+2, etc., we expect

$$\begin{aligned} p_{i+1} &= p_i M = \begin{bmatrix} 2/3 & 1/3 \end{bmatrix} \begin{bmatrix} 2/3 & 1/3 \\ 1/3 & 2/3 \end{bmatrix} = \begin{bmatrix} 0.55556 & 0.44444 \end{bmatrix} \\ p_{i+2} &= p_{i+1}M = [0.51852 \;\; 0.48148] \\ &\ldots\ldots \\ p_{i+9} &= p_{i+8}M = [0.50001 \;\; 0.49999] \end{aligned} \tag{6.5}$$

If we continue the multiplication, eventually the two frequencies will approach $p_R = p_Y = 0.5$. These equilibrium frequencies can be obtained more easily by solving the equation $p_i = p_{i-1}M$ with the constraints that

$$\begin{aligned} &p_{R.i} = p_{R.i-1} \\ &p_{Y.i} = p_{Y.i-1} \\ &p_{R.i} + p_{Y.i} = 1 \end{aligned} \qquad (6.6)$$

Solving the resulting simultaneous equations, we have

$$p_{R.i} = \frac{P_{YR}}{P_{RY} + P_{YR}};\ p_{Y.i} = \frac{P_{RY}}{P_{RY} + P_{YR}} \qquad (6.7)$$

which are the equilibrium frequencies. The equilibrium frequencies are conventionally designated as π_R and π_Y, respectively. For the fictitious sequence S, $\pi_R = p_R = 0.5$ and $\pi_Y = p_Y = 0.5$. For human chromosome 22, $\pi_R = p_R = 0.50047$ and $\pi_Y = p_Y = 0.49953$.

One can use one of many numerical routines, e.g., the Gauss-Jordan elimination (Press *et al.*, 1992, pp. 36-43) to solve for π_i by following these steps: (1) change the diagonal elements of M to $-(1-P_{ii})$, (2) transpose M to M^T, (3) create a vector B with n elements all being 0, (4) add the constraint of $\Sigma p = 1$ by adding 1 to the first row of M, and by setting B(0) = 1, and (4) use any numerical routine that solves $M^Tp = B$.

You may ask whether 1st-order Markov model is any better than the 0-order Markov model. The latter assumes complete independence of nucleotides over sites, i.e., whether site i is R has nothing to do with what nucleotides are at other sites. Because the 0-order Markov model is a special case of the 1st-order Markov model, the question can be addressed by what is called a likelihood ratio test. This naturally leads to the calculation of the likelihood.

Likelihood refers to either a probability function conditional on a probabilistic model and the empirical observation or the result of such a function. For the fictitious sequence S' (the empirical observation) with only 12 nucleotides, the likelihoods according to the 0-order and 1st-order Markov models are, respectively:

$$\begin{aligned} L_0 &= p_R^3 p_Y^3 p_R^3 p_Y^3 = 0.000244141 \\ L_1 &= p_R P_{RR}^2 P_{RY} P_{YY}^2 P_{YR} P_{RR}^2 P_{RY} P_{YY}^2 = 0.000722564 \end{aligned} \qquad (6.8)$$

The larger the likelihood, the better the model fits the data. However, the absolute magnitude of a likelihood value is not important, what is important is the relative magnitude measured as a ratio of two likelihood values. In our case, what is important is the ratio of L_1/L_0 which can be used in a likelihood ratio test to help us decide whether the 1st-order Markov model is significantly better than the 0-order Markov model. To carry out a likelihood ratio test, we calculate,

$$X^2 = 2[\ln(L_1) - \ln(L_0)] = 2.170122511 \qquad (6.9)$$

which follows approximately the χ^2-distribution with one degree of freedom. The degree of freedom associated with a likelihood ratio test is the difference in the number of parameters between the two models. The 0-order Markov model has only two frequencies, p_R and p_Y. Because $p_R = 1- p_Y$, it is not free when p_Y is known, so there is only one free parameter for the 0-order Markov model. For the 1st-order Markov chain, we have four elements in the transition probability matrix M. However, because the two values in each row have to add up to 1, M has only two free parameters. Thus, the degree of freedom associated with the likelihood ratio test involving the 1st-order and 0-order Markov models is (2 – 1) = 1. In general, for a kth-order Markov model with n categories of symbols (in our case we have only two categories of symbols, i.e., R and Y, so n = 2), the number of parameters is

$$N_{param} = (n-1)n^k \qquad (6.10)$$

For example, the number of parameters for a 3rd-order Markov model for an amino acid sequences (n = 20) is $19\times 20^3 = 15200$. Given that a protein is generally only one thousand amino acids long, such a model may have only limited use.

The likelihood ratio test does not reject the 0-order Markov model in favour of the 1st-order Markov model for the given S' ($p = 0.14$), although the site dependence, with purine triplets and pyrimidine triplets following each other, is quite obvious. The reason for the failure to reject the 0-order Markov model is the short length of S'. If S' is three times longer, i.e., if it is made of the first 36 nucleotides of S, then $X^2 = 4.888507099$ and we can reject the 0-order Markov model with $p = 0.027036057$.

P_R and P_Y from S happen to be the same as π_R and π_Y. Usually P_R and P_Y from a short sequence will not be the same as π_R and π_Y. If one has already obtained π_R and π_Y from longer sequences, one should replace P_R with π_R in Eq. (6.8).

3. HIDDEN MARKOV MODELS

3.1 The Essential Elements in a Hidden Markov Model

The classic illustration of HMM involves a dishonest casino dealer stealthily switching between a fair die (F) and a loaded die (L). His switching pattern is characterized by the 2×2 transition probability matrix (Table 6-2) detailed in the previous section. Now we introduce another kind of probability called emission probability. Tossing the fair die leads to numbers 1-6 appearing with an equal probability of 1/6, but tossing the loaded die always results in number 6. These probabilities of observing different numbers conditional on the hidden state are termed emission probabilities (Table 6-2).

Table 6-2. HMM involving a dishonest casino dealer, with the transition probability matrix on the left and the emission probabilities on the right (the last 6 columns).

	F	L	1	2	3	4	5	6
F	P_{FF}	P_{FL}	1/6	1/6	1/6	1/6	1/6	1/6
L	P_{LF}	P_{LL}	0	0	0	0	0	1

Because the casino dealer will not let others know when he switches, the two states (F and L) are hidden. Hence the term "hidden Markov model". What one can observe is just a series of numbers, e.g.,

```
Observed symbols: 1435266634521334
Hidden states:    FFFFFLLLFFFFFFFF
```

To summarize, a HMM has three essential elements, the transition probability matrix, the emission probability matrix, and the observed sequence of events. Recall that signal sensors represent information of site dependence along a sequence. The transition probability matrix is a mathematical instrument making use of signal sensors. If there is little site dependence, then the transition probability matrix will be of little use.

In contrast, content sensors represent word frequencies in a sequence without site-dependence. The emission probability matrix is a mathematical instrument making use of content sensors. Given a hidden states, the emission probabilities are always the same regardless of where the hidden state is located on the sequence of observed symbols. If there is no site dependence among the sequence of hidden states and if emission probabilities do not differ among different hidden states, then the HMM would be entirely uninformative.

Another illustrative application of HMM is in base-calling (the process of converting the four traces from the automatic sequencer to nucleotide sequences). In this case the traces are the observed signals, and the bases are the hidden states. More advanced applications of HMM can be found in phylo-HMM which is a site-dependent probabilistic substitution model supposed to be a better approximation to the substitution process in sequence evolution (Felsenstein and Churchill, 1996; Siepel and Haussler, 2004a, 2004b, 2005; Yang, 1995).

The application of HMM in base-calling is probably not a very well thought one. Typically, the association between a base and a particular peak is strong, i.e., a given hidden state has a strong and characteristic peak. In other words, emission probability matrix is highly informative and base-calling can generally be quite successful with this emission probability matrix only. In contrast, the site dependence of neighbouring nucleotides is generally weak and generally does not significantly improve the prediction.

HMMs are generally associated with three objectives. The first is to estimate the parameters of HMM, e.g., the elements in the transition probability matrix and in the emission probability matrix, based on an observed sequence of events with known hidden states. This is also called HMM training. The second is to reconstruct the most probable path of hidden states by using the trained HMM and the Viterbi algorithm explained in detail later. The last is to obtain the probability of the observed sequence of events, also explained in detail later.

How is HMM associated with gene prediction and motif finding? One may consider exons, introns, etc., as hidden states, and a series of nucleotide triplets along a genomic sequence as observed symbols. The frequencies of different triplets "emitted" in the exon state are often different from those in the intron state. Furthermore, an intron is always followed by an exon, so the dependence of neighboring states is obviously strong. These different emission probabilities and transition probabilities between neighboring hidden states allow us to reconstruct the hidden states of exons and introns. Another example of HMM application is in predicting protein secondary structural elements such as α-helix, β-sheet and turns. These structural elements are hidden states that emit different amino acids with different frequencies. For example, some amino acids are frequently found in helices but rarely in sheets or turns (Xia and Xie, 2002).

Whether a HMM is successful depends crucially on two things. The first is how different are the emission probabilities among different states. Take the dishonest casino dealer for example. If the loaded die is loaded only slightly, then it will be extremely difficult for us to reconstruct the path of hidden states. If the observable nucleotide, dinucleotide and trinucleotide frequencies (or other observable symbols) are not very different between coding and non-coding sequences, then the two would be difficult to tell apart. For this reason, a purely computational scientist without access to in-depth knowledge of the differences among different kinds of sequences is unlikely to make a successful application of HMM in gene and motif prediction. The second is how long the process will stay in each state. For example, if the dishonest casino dealer throws the loaded die several times consecutively, then a series of 6's will help us identify the hidden states. However, if he never throws the loaded die more than once, then it will be essentially impossible for us to reconstruct the hidden states even if the loaded die always yields a six. This means that short exons, which are frequently encountered in eukaryotic genomes, will be very difficult to identify.

In what follows, I will use empirical data from the HIV1 proteins to illustrate the application of HMM in secondary structure prediction. In short, we will learn to (1) train a HMM, i.e., obtain the transition probability matrix M and emission probability matrix E, from a training sequence with known hidden states, (2) reconstruct the sequence of hidden states, known as the Viterbi path or the most probable path, by using the Viterbi algorithm (Viterbi, 1967), and (3) compute the probability of the observed sequence of symbols. These are the three essential tasks closely associated with HMM (Rabiner, 1989). The last two tasks require the transition probability matrix and the emission probability matrix from task 1.

3.2 Training HMM

Suppose we are going to use a training sequence (Figure 6-1) to train a HMM for predicting protein secondary structure defined to be either coil (C), strand (E) or helix (H). The sequence labelled as RT is a protein sequence of HIV1 reverse transcriptase, and the sequence labelled as ST is the sequence of known hidden states. Let N_{ij} be the number of transitions from state i to state j, which can be easily counted by moving along the ST sequence (Figure 6-1). The maximum likelihood estimate of P_{ij}, i.e., the transition probability from state i to state j, is

$$P_{ij} = \frac{N_{ij}}{\sum_k N_{ik}} \tag{6.11}$$

The P_{ij} values from data in Figure 6-1, with relatively large P_{ii} values on the diagonal (Table 6-3) are typical of training data where each secondary structure elements are made of a series of consecutive amino acids so that a state, be it C, E, or H, tends to stay in the same state for several consecutive amino acids before changing into another state (Figure 6-1). Note that each row sum should be 1.

```
ST CCCCCCCEEEEECCCCCCCCCCCCCCCHHHHHHHHHHHHHHHHCCCEEEECCC
RT PISPIETVPVKLKPGMDGPKVKQWPLTEEKIKALVEICTEMEKEGKISKIGPE

ST CCCCCCEEEEEECCCCCHHHHHHHHHHHHHHHHHHEEECCCCCCCCCCCCCCE
RT NPYNTPVFAIKKKDSTKWRKLVDFRELNKSTQDFWEVQLGIPHPAGLKKKKSV

ST EEEEECCEEEECCCCCCCCEECCEECCCCCCCCCCCCCEECCCCCCCCCCHHH
RT TVLDVGDAYFSVPLDEDFRKYTAFTIPSINNETPGIRYQYNVLPQGWKGSPAI

ST HHHHHHHHHHHHHHCCCCEEEEEEECCCCCCCCCCHHHHHHHHHHHHHHHHHH
RT FQSSMTKILEPFRKQNPDIVIYQYMDDLYVGSDLEIGQHSTKIEELRQHLLRW

ST CCCCCCCCCCCCCCCCCCCCEECCCCCEECCEECCCCCCCCHHHHHHHHHCCC
RT GLTTPDKKHQKEPPFLWMGYELHPDKWTVQPIVLPEKDSWTVNDIQKLVGKLN

ST HHHHHCCCCHHHHHHHHHCCCCCCCCCCCCCHHHHHHHHHHHHHHHCCCCCEE
RT WASQIYPGIKVRQLCKLLRGTKALTEVIPLTEEAELELAENREILKEPVHGVY

ST ECCCHHHHHHHHHCCCCCEEEEEECCCCCCCCCCCCCCCCCCCHHHHHHHHHH
RT YDPSKDLIAEIQKQGQGQWTYQIYQEPFKNLKTGKYARMRGAHTNDVKQLTEA

ST HHHHCCCCEEEECCCCCCCCCCHHHHHHHHHHHHHHHCCCCCCCCCCCCCCCC
RT VQKITTESIVIWGKTPKFKLPIQKETWETWWTEYWQATWIPEWEFVNTPPLVK

ST EEEEECCCCCCCCCCC
RT LWYQLEKEPIVGAETF
```

Figure 6-1. Example of a training sequence (RT) with known hidden states (ST) for protein secondary structure prediction. The hidden states are coil (C), strand (E) and helix (H). RT is a real HIV1 reverse transcriptase. ST is fictitious but treated here as real.

Table 6-3. Transition probability matrix estimated from the training data in Figure 6-1.

	C	E	H
C	0.88210	0.06987	0.04803
E	0.26154	0.73846	0.00000
H	0.06897	0.00690	0.92414

For the emission probabilities, letting $E_k(x_j)$ be the number of amino acid x_j ($j = 1, 2, \ldots, 20$ corresponding to the 20 amino acids) emitted from state k, the emission probability $e_k(x_j)$ is

$$e_k(x_j) = \frac{E_k(x_j)}{\sum_{j=1}^{20} E_k(x_j)} \tag{6.12}$$

Note that the denominator is the sum of all amino acid in state k, and the numerator is the number of amino acid x_j in state k. So $e_k(x_j)$ is simply the fraction of amino acid x_j in state k.

The $e_k(x_j)$ values estimated from the training data in Figure 6-1 reveal different amino acid frequency distribution among the three secondary structure features (Table 6-4). For example, amino acids proline (P) and glycine (G) are found frequently in coils, but rarely in strands or helices. This should be obvious given the properties of proline and glycine. Proline introduces a bend in protein structure due to the cyclic binding of its three-carbon side chain to the nitrogen of the backbone. Glycine is the smalles amino acid, and only a small amino acid allows a sharp turn.

Table 6-4. Emission probabilities estimated from the training data in Figure 6-1. The column heading "AA" stands for amino acid, and C, E, and H stands for coil, strand, and helix, respectively.

AA	C	E	H
A	0.0262	0.03077	0.05517
C	0	0	0.01379
D	0.05677	0.01538	0.03448
E	0.07424	0.03077	0.13793
F	0.02183	0.04615	0.02759
G	0.09607	0	0.01379
H	0.02183	0	0.01379
I	0.04803	0.13846	0.08276
K	0.11354	0.07692	0.11724
L	0.06987	0.07692	0.11034
M	0.0131	0.01538	0.01379
N	0.0393	0	0.02759
P	0.13974	0.01538	0.01379
Q	0.0393	0.07692	0.08276
R	0.02183	0	0.06207
S	0.0393	0.03077	0.02069
T	0.08297	0.06154	0.06207
V	0.0393	0.18462	0.04828
W	0.03057	0.04615	0.05517
Y	0.0262	0.15385	0.0069

Two other patterns in Table 6-4 are worth highlighting. Tyrosine (Y) and valine (V) are frequently found in strands but not in the other two structures, and leucine (L) and glutamate (E) are frequently found in helices but rare elsewhere (Table 6-4 and Figure 6-1). These differences in emission probabilities are encouraging. The larger the difference, the better the HMM will perform in using the emission probabilities and transition probabilities to predict secondary structures. If the three columns of frequencies in Table 6-4 are similar, then the predictive power of the HMM will rest entirely on the information in the transition probability matrix. It is crucially important for computational biologists to collaborate with real molecular biologists to define the HMM model structure to have hidden states with maximal differences in their emission probabilities.

3.3 The Viterbi algorithm

Now that we have gone through the training process of obtaining the transition probability matrix (Table 6-3) and the emission probabilities (Table 6-4), we are ready to proceed with the prediction of protein secondary structure of unknown proteins with the Viterbi algorithm illustrated in this section. Because there are overlapping characters between the secondary structure notation (C, E and H) and the one-letter codes of amion acids, I will add the full notation in parenthesis to avoid confusion.

Suppose we have the following amino acid sequence:

```
T = YVYVEEEEEEVEEEEEEPGPG
```

How do we predict its secondary structure? Of course we can use only the content sensor as reflected by the emission probabilities (Table 6-4). Given the association of L (leucine) and E (glutamate) with helix, P (proline) and G (glycine) with coil (C) and Y (tyrosine) and V (valine) with strand (E), we can readily write our prediction of the secondary structure referred to as the naïve path of hidden states (Naïve):

```
          123456789012345678901
T       = YVYVEEEEEEVEEEEEEPGPG
Naïve   = EEEEHHHHHHEHHHHHHCCCC
```

This seems to make secondary structure prediction really easy, and the approach has actually been taken before (Chou and Fasman, 1978b, 1978a; Fasman and Chou, 1974). However, incorporating information on site-dependence can improve the prediction. For example, P_{EH} in the transition probability matrix is 0.00000, implying an extremely small probability of E (strand) followed by H (helix). Our naïve prediction above with an H (helix) at position 5 following an E (strand) at position 4 therefore represents an extremely unlikely event. Another example is at position 11 with T_{11} = V (valine). Our prediction of $Naïve_{11}$ = E (strand) implies a transition of secondary structure from H (helix) at $Naïve_{10}$ to E (strand) at $Naïve_{11}$ and then back from E (strand) at $Naïve_{11}$ to H (helix) at $Naïve_{12}$. The transition probability matrix shows us that P_{HE} and P_{EH} are both very small. So T_{11} is very unlikely to be in state E (strand).

Let us see if the Viterbi algorithm in HMM can do better than the naïve prediction. The Viterbi algorithm (Viterbi, 1967) is a dynamic programming algorithm. It incorporates both the information in the transition probability matrix and the emission probability matrix. The computation involves filling a Viterbi matrix and a backtrack matrix (referred to as V and B, respectively, hereafter), both of dimension n×L where n is the number of states and L is the length of the sequence. In our example, n = 3 and L = 21. Because the page is not wide enough for 21 columns, V and B are shown in L×n tables (Table 6-5).

Table 6-5. The V and B matrices from running the Viterbi algorithm using the transition probability matrix in Table 6-3 and emission probability matrix in Table 6-4. The values in the V matrix are their natural logarithms. The values in the B matrix are pointers, with 0, 1, and 2 as pointers to C, E, or H state in the previous site. The last column (HS) is the reconstructed hidden states.

	V Matrix			B Matrix			
	C	E	H	C(0)	E (1)	H(2)	HS
Y	-4.74057	-2.97041	-6.07535				E
V	-7.54809	-4.96308	-9.18506	1	1	2	E
Y	-9.94622	-7.13807	-14.24069	1	1	2	E
V	-11.71574	-9.13074	-16.01287	1	1	0	E
E	-13.07242	-12.91516	-16.73257	1	1	0	C
E	-15.79838	-16.69959	-18.08925	0	1	0	H
E	-18.52435	-20.48402	-20.14914	0	1	2	H
E	-21.25031	-24.26844	-22.20904	0	1	2	H
E	-23.97627	-27.39268	-24.26893	0	0	2	H
E	-26.70223	-30.11864	-26.32883	0	0	2	H
V	-30.06419	-31.05285	-29.43855	0	0	2	H
E	-32.79015	-34.83727	-31.49844	0	1	2	H
E	-35.51611	-38.62170	-33.55834	0	1	2	H
E	-38.24207	-41.65849	-35.61823	0	0	2	H
E	-40.89289	-44.07621	-37.67813	2	2	2	H
E	-42.95279	-46.13610	-39.73802	2	2	2	H
E	-45.01268	-48.19600	-41.79792	2	2	2	H
P	-46.44005	-50.94904	-46.16040	2	2	2	C
G	-48.90819	-237.91316	-50.52288	0	0	2	C
P	-51.00163	-55.74371	-54.88536	0	0	2	C
G	-53.46976	-242.47474	-58.32104	0	0	0	C

The output in Table 6-5 is from program DAMBE (Xia, 2001; Xia and Xie, 2001b) which implements the Viterbi algorithm as well as the forward algorithm (detailed later) for computing the probability of the observed sequence of events given the transition probability matrix and the emission probability matrix. Here we use manual computation so that you know how to get the output yourself.

The first amino acid Y has no site-dependence information because we do not know what goes before it. So the three values in the first row of V (Table 6-5) are filled as a function of emission probabilities. Given amino acid Y and no other information, the likelihood of the hidden states being C, E and H are respectively $e_C(Y)$, $e_E(Y)$ and $e_H(Y)$ which are 0.0262, 0.15385 and 0.0069, respectively, from Table 6-4. These three values are divided by the number of hidden states to give

$$\begin{aligned} V_C(1) &= 0.0087336245 \\ V_E(1) &= 0.0512820513 \\ V_H(1) &= 0.0022988506 \end{aligned} \tag{6.13}$$

You may wonder why these three values are not the same as the three values in the first row in the V matrix in Table 6-5. This will be explained soon.

In more concise and more general mathematical terms, Eq. (6.13) is written as

$$V_k(1) = e_k(X_1)/n \tag{6.14}$$

where k = C, E, or H, and n is the number of hidden states (=3 in our example).

An alternative initialization of $V_k(1)$ values is

$$V_k(1) = e_k(X_1)\pi_{X_1} \tag{6.15}$$

where π_{X1} is the equilibrium frequency of X_1. We have already learned how to compute the equilibrium frequencies in a previous section on 1st-order Markov models. This way of initialization is numerically illustrated later when we learn the forward algorithm.

The rest of the rows in V are filled with a more intimidating equation

$$V_l(i) = e_l(X_i)\max_k(V_k(i-1)P_{kl}) \tag{6.16}$$

where subscript *l* refers to the hidden state at site *i*, and *k* is the hidden state at site *i*-1, P_{kl} is the transition probability from hidden state *k* to hidden state *l*.

Many intimidating mathematical expressions are in fact quite simple, and Eq. (6.16) is no exception, especially when it is rendered into numbers. For example, according to the equation, the second row of the V matrix is filled with the following three values:

$$\begin{aligned}
V_C(2) &= e_C(V)\max[V_C(1)P_{CC}, V_E(1)P_{EC}, V_H(1)P_{HC}] \\
&= 0.0393\times\max(0.008733625\times 0.8821, \\
&\quad 0.051282051\times 0.26154, 0.002298851\times 0.06897) \\
&= 0.0393\times 0.013412308 = 0.000527104 \\
V_E(2) &= e_E(V)\max[V_C(1)P_{CE}, V_E(1)P_{EE}, V_H(1)P_{HE}] \\
&= 0.006991512 \\
V_H(2) &= e_H(V)\max[V_C(1)P_{CH}, V_E(1)P_{EH}, V_H(1)P_{HH}] \\
&= 0.000102569
\end{aligned} \tag{6.17}$$

where V in $e_C(V)$, $e_E(V)$ and $e_H(V)$ is the amino acid V at the second site of the amino acid sequence. Don't confuse it with the V matrix.

$V_l(i)$ values for i = 3, 4, …, 21 are computed in exactly the same way. Each one depends on the $V_l(i-1)$ values. This is typically of all dynamic programming algorithms.

Eq. (6.16) and its numerical rendition in Eq. (6.17) are also very easy to understand and to remember. Take $V_C(2)$ in Eq. (6.17) for example. The max function is to find, given the reconstructed hidden state at the second site is C, which of the three possible hidden states in the previous site is most likely to transit into C. In our case, it is the hidden state E that is most likely to transit into C, with its value of 0.013412308 being the maximum of the three. If the reconstructed hidden state is C, how likely is it to emit an amino acid V? This is the emission probability $e_C(V)$ = 0.0393 (Table 6-4). So $V_C(2)$ is the probability that the hidden state at the site 2 is C multiplied by the probability that the hidden state C emits an amino acid V. With this, the computation of $V_E(2)$ and $V_H(2)$, as well as the rest of $V_l(i)$, is obvious. I encourage you to perform the computation by hand. Just in case you wish to have some values to check your computation, here are the three $V_l(3)$ values:

$$\begin{aligned} V_C(3) &= 0.0000479085 \\ V_E(3) &= 0.0007942838 \\ V_H(3) &= 0.0000006537 \end{aligned} \tag{6.18}$$

Now you may have a really burning question in your mind. The three rows of V matrix that we have computed in Eq. (6.13), Eq. (6.17) and Eq. (6.18) are supposed to be the same as the first three rows in the V matrix in Table 6-5. Why are they not the same? Please be patient for just one more minute.

You might have noticed that the three $V_l(2)$ values in Eq. (6.17) are substantially smaller than the three $V_l(1)$ values in Eq. (6.13), and the three $V_l(3)$ values in Eq. (6.18) are substantially smaller than the three $V_l(2)$ values in Eq. (6.17). The Viterbi algorithm involves the multiplication of many small probabilities and it takes only a short sequence for a computer to generate an arithmetic underflow or overflow error (and probably weird results long before this). Do you have a solution for this problem now that you understand how to compute the V matrix?

The solution turns out to be simple. In order to avoid arithmetic underflow or overflow problems in computation, any practical implementation of the Viterbi algorithm would have log-transformed the equations so that we do additions instead of multiplications. If you take the natural logarithm of $V_l(1)$, $V_l(2)$ and $V_l(3)$ values in equations (6.13), (6.17), and (6.18), you should get the values in the first three rows in the V matrix in Table 6-5.

We have not talked about the B matrix (the backtrack matrix), which should be filled concurrently with the V matrix. Just as the backtrack matrix in the dynamic programming algorithm for sequence alignment is for the actual reconstruction of aligned sequences, the backtrack matrix in the Viterbi algorithm is for reconstructing the most probable hidden path, also known as the Viterbi path. In our example, the Viterbi path is the reconstructed secondary structure of the peptide T.

Each cell in B in Table 6-5 is in fact a pointer (or arrow) to a cell in the previous site. The first row of B in Table 6-5 is empty for the obvious reason that the first site has no previous site to point to (Table 6-5). The first three values in the second site is 1, 1, and 2 (Table 6-5). We will first explain how we get these values and what they mean, and finally learn how to reconstruct the Viterbi path by following the pointers in the B matrix.

The entries in the B matrix are obtained from the max function in equations (6.16) and (6.17). Take $V_C(2)$ in Eq. (6.17) for example. The three values within the max function are $V_C(1)P_{CC}$, $V_E(1)P_{EC}$ and $V_H(1)P_{HC}$, with the maximum being $V_E(1)P_{EC}$. So we should have an arrow pointing from C at the second site to E at the first site. Because it is not convenient to store graphic arrows in digital computers, we coded the three hidden states C, E and H to 0, 1, and 2, respectively. To represent an arrow from C at the second site to E at the first site, we simply put a number 1 under column C (B matrix in Table 6-5) at the second site. Because we only have three hidden states, a cell in B will contain either a 0 (for C) or 1 (for E) or 2 (for H).

Yes, I could have put real arrows to the B matrix in Table 6-5, but I was afraid of spoiling future programmers if everything is made too visual. Sometimes it is better to see things with our mind's eye. However, there is nothing preventing you from replacing the numbers by arrows in the B matrix in Table 6-5.

The second value in the second row of B is 1, meaning an arrow pointing from state E at the second site to state E at the first site (i.e., a vertical arrow pointing up). This is obtained in exactly the same way from $V_E(2)$ in Eq. (6.17). Among the three values within the matrix function, $V_C(1)P_{CE}$, $V_E(1)P_{EE}$, and $V_H(1)P_{HE}$, $V_E(1)P_{EE}$ is the largest. So we again have a value of 1 representing an arrow from state E at the second site to state E at the first site.

For $V_H(2)$ in Eq. (6.17), the last of the three values within the max function is the largest, yielding a value of 2 representing an arrow from state H in the second site to H in the first site. That is, you have another vertical arrow pointing upwards.

Once the V and B matrices are complete, we can backtrack along the B matrix to reconstruct the hidden states. We first look at the very last row in V (Table 6-5) and find the largest value, which is -53.46976 under column C. This means that the last amino acid (i.e., G) should be in state C (i.e., in a coil). This brings us to the value in the last row of B under column C in Table 6-5. This value is 0 (representing C). Recall that values in the B matrix are pointers. A value of 0 in a cell in the B matrix means that the second last amino acid (i.e,. P) is also in a coil. Similarly, we know the third and fourth last amino acids (G and P, respectively) are also in a coil. We record these reconstructed hidden states in the last column in Table 6-5, i.e., a stretch of four C's from the bottom.

Now we are at the cell containing a value of 2 (for H or helix) under column C (the fourth row from the bottom). This means that the next amino acid (i.e., amino acid E at site 17) is no longer in state C but in state H. So now we move to the value of 2 (the fifth from the bottom) under column H in Table 6-5. This cell, together with the 11 cells in proceeding sites, contains a value of 2. This means a stretch of amino acids in state H all the way to the 6th amino acid (i.e., amino acid E). We again record this stretch of H's in the last column (HS) in Table 6-5.

The last value of 2 in this stretch of 2's is in the 7th row of the B matrix in Table 6-5, corresponding to amino acid E. The cell right above this 2 is 0. This means that the amino acid at 5th site (i.e., E) is no longer in state H, but in state C. This brings us back to column C corresponding to the fifth amino acid (amino acid E), where we find a value of 1. This value of 1 means that the previous amino acid (the fourth one, amino acid V) is in structure E. This brings us to column E at the forth amino acid. This cell has a value of 1 and a stretch of 1's above it all the way to the top. This means that the four amino acids at sites 1-4 are all in structure E. If you find yourself confused, then just replace those numbers in the B matrix in Table 6-5 by real arrows and then follow the arrows to get the reconstructed secondary structure.

The final reconstructed sequences are shown below (Viterbi), together with the naïve reconstruction (Naïve) we have derived by using only information in the emission probability matrix:

```
           123456789012345678901
T        = YVYVEEEEEEVEEEEEEEPGPG
Viterbi  = EEEECHHHHHHHHHHHHCCCC
Naïve    = EEEEHHHHHHEHHHHHHCCCC
```

Two hidden states, at site 5 and 11, were reconstructed differently between the naïve path and the Viterbi path. A hidden state of H at site 5 in the Naïve reconstruction implies a transition from hidden state E directly into hidden state H, which has an extremely small probability P_{EH} = 0.00000 (Table 6-3). The Viterbi path shows first a transition from E to C and then from C to H. This is a more likely path than the Naïve reconstruction because P_{EC} and P_{CH} are both much larger than 0.00000 (Table 6-3). Another difference is at site 11. The naïve reconstruction of state E at this site implies a transition of secondary structure from H to E and then back from E to H. The transition probability matrix shows us that P_{HE} and P_{EH} are both very small. So a hidden state of E at this site is very unlikely. The transition probability matrix shows a large P_{HH} value (Table 6-3). The reconstructed hidden state H at this site in the Viterbi path implies that the helix structure is more likely to continue across this site instead of switching to some other secondary structures.

We have now covered two of the three major tasks associated with HMM, i.e., train a HMM and reconstruct the sequence of hidden states. We now deal with the last task, i.e., computing the probability of the observed sequence of symbols by the forward algorithm (Rabiner, 1989).

3.4 Forward algorithm

The forward algorithm is for computing the probability of the observed sequence of events. Given an observed amino acid sequence T, the probability is designated P(T). This probability is useful to understand the process generating the sequence. For example, given the transition probability matrix and the emission probability matrix, if we find the observed sequence (say amino acid sequence T) to have a much smaller probability than any of the training sequences of the same length from the training set, then we may have to conclude either that the training set is not representative (i.e.., the sequence T should not be a member of the training set) or that the HMM structure is wrong, i.e., there might be more hidden states or the hidden states in the training set might be wrongly assigned. HMM is often used to decide whether an amino acid sequence belongs to a protein family and P(T) plays an important role in making such a decision. In such cases, the HMM is derived from a set of aligned protein sequences known to belong to a particular protein family. We classify a new sequence T into the protein family when P(T) is large.

The forward algorithm involves dynamically completing a L×n matrix, where L is sequence length and n is the number of hidden states (Table 6-6). The matrix is henceforth referred to as the F matrix (F for forward).

Table 6-6. The F matrix from the forward algorithm. The first column is the amino acid sequence (AA), and the last column is the scaling factor s(i) explained in the text.

AA	C	E	H	s
Y	0.35294	0.58824	0.05882	25.824
V	0.17282	0.79491	0.03226	9.371
Y	0.09317	0.90426	0.00258	9.807
V	0.09124	0.90635	0.00241	7.281
E	0.52078	0.45909	0.02013	22.082
E	0.71047	0.19040	0.09914	16.477
E	0.68601	0.07944	0.23455	13.523
E	0.55790	0.03897	0.40313	11.704
E	0.40736	0.02247	0.57018	10.351
E	0.28088	0.01410	0.70502	9.353
V	0.23427	0.12798	0.63775	19.865
E	0.19622	0.03300	0.77078	9.304
E	0.14736	0.01129	0.84135	8.452
E	0.11512	0.00610	0.87878	8.120
E	0.09685	0.00456	0.89858	7.968
E	0.08706	0.00397	0.90898	7.892
E	0.08192	0.00369	0.91439	7.853
P	0.61471	0.00733	0.37797	32.278
G	0.91292	0.00000	0.08708	16.665
P	0.97669	0.00853	0.01477	8.615
G	0.99004	0.00000	0.00996	11.917

The three values in the first row, corresponding to the first amino acid (i.e., Y) are filled as a function of emission probabilities, exactly as the V matrix in the Viterbi algorithm. Given Y and no other information, the likelihood of the hidden states being C, E and H are respectively $e_C(Y)$, $e_E(Y)$ and $e_H(Y)$ which are 0.0262, 0.15385 and 0.0069, respectively, from Table 6-4. One can initialize the first three values by dividing these three values by the number of hidden states (= 3) as in Eq. (6.14) or by multiplying these values by the equilibrium frequencies of the hidden states as in Eq. (6.15). The three values from the first method of initialization are equal to $V_l(1)$ in Eq. (6.13). To learn something new, we will use the second method of initialization. The equilibrium frequencies for hidden states C, E, and H can be computed by using the method explained in a previous section on Markov models and are equal to 0.52164009, 0.14806378, and 0.33029613, respectively. This yields

$$\begin{aligned}F_C(1) &= 0.01366697\\ F_E(1) &= 0.022779613\\ F_H(1) &= 0.002279043\end{aligned} \tag{6.19}$$

You may begin to wonder why the three $F_l(1)$ values are not the same as the three values in the first row in the F matrix in Table 6-6. It may suddenly dawn on you that you need to take their logarithms. So you did, and …… they are still quite different. You may begin to wonder if I have made computational errors. Rest assured that I did not. You just need to be a bit more patient.

The $F_l(i)$ values with i > 1 are computed according to the following equation

$$F_l(i+1) = e_l(X_{i+1})\sum\nolimits_k F_k(i)P_{kl} \tag{6.20}$$

Again, this seemingly intimidating equation is quite easy to understand and simple to compute, especially when it is rendered to numbers. Take the second amino acid site (amino acid V) for example. The three $F_l(2)$ values are:

$$\begin{aligned}
F_C(2) &= e_C(V)(F_C(1)P_{CC} + F_E(1)P_{EC} + F_H(1)P_{HC}) \\
&= 0.0393(0.01366697 \times 0.8821 \\
&\quad +0.022779613 \times 0.26154+0.002279043 \times 0.06897) \\
&=0.000714105 \\
F_E(2) &= e_E(V)(F_C(1)P_{CE} + F_E(1)P_{EE} + F_H(1)P_{HE}) \\
&= 0.003284846 \\
F_H(2) &= e_H(V)(F_C(1)P_{CH} + F_E(1)P_{EH} + F_H(1)P_{HH}) \\
&= 0.000133377
\end{aligned} \tag{6.21}$$

We continue until we obtain the three values for the last amino acid at site 21 (i.e., the terminating G). The summation of these last three values is the probability of the observed amino acid sequence (T) given the transition probability matrix and the emission probability matrix, i.e.,

$$P(T) = \sum_{l=1}^{n} F_l(L) \tag{6.22}$$

where *l* is the index of hidden states (E, C and H in our example), n is the number of hidden states (= 3 in our example) and L is the sequence length of T. However, Eq. (6.22) is almost never used in actual computation because, without rescaling, $F_l(L)$ will often be too small to be represented in digital computers with a large L. The actual computed P(T) in our example is 3×10^{-23}, a very small value. In practice, we always compute the natural logarithm of the probability because the probability itself will be difficult to represent in digital computers with a large L. The natural logarithm of the probability is -51.8606178.

You might again wonder why the three $F_l(1)$ values in Eq. (6.19) and the three $F_l(2)$ values in Eq. (6.21) are not the same as the six values populating the first two rows of F in Table 6-6. The reason is that the forward algorithm involves the multiplication of many small probabilities and some computational tricks have to be taken to avoid the arithmetic underflow or overflow error. You might suggest using the logarithm as we did before with the Viterbi algorithm, but this time the logarithm does not help because of the summation term in Eq. (6.20). What we typically do is to re-scale the $F_l(i)$ values by a scaling factor s(i). In DAMBE (Xia, 2001; Xia and Xie, 2001b), the three Fl(i) values are re-scaled to

$$F_l^{'}(i) = s(i)F_l(i);$$
$$s(i) = \frac{1}{\sum_{l'} F_{l'}(i)} \qquad (6.23)$$

For example, the three $F_l(1)$ values are re-scaled to

$$F_C^{'}(1) = \frac{F_C(1)}{F_C(1) + F_E(1) + F_H(1)} = 0.352917995$$
$$F_E^{'}(1) = 0.588230967 \qquad (6.24)$$
$$F_H^{'}(1) = 0.058851038$$

It is these re-scaled values that are presented in Table 6-6. The last column in Table 6-6 is the scaling factor in Eq. (6.23) that can be used to compute P(T):

$$P(T) = \frac{1}{\prod_{i=1}^{L} s_i}$$
$$\ln[P(T)] = -\sum_{i=1}^{L} \ln(s_i) \qquad (6.25)$$

where L is the sequence length.

3.5 HMM and gene prediction

Many methods have been used for gene prediction in prokaryotic genomes (Borodovsky and McIninch, 1993a; Krogh *et al.*, 1994; Salzberg *et al.*, 1998) and eukaryotic genomes (Besemer and Borodovsky, 2005; Burge and Karlin, 1997; Burge and Karlin, 1998). The application of HMM in gene prediction involves defining the structure of HMM with selected hidden states, training the defined HMM with genomic sequences of known hidden states to estimate the parameters in the transition probability matrix and the emission probability matrix, and finally apply the method to predict genes (i.e., reconstruct hidden states) by using the Viterbi and forward algorithms.

It is not trivial to define the model structure of HMM in gene prediction. Current methods for gene prediction in eukaryotic genomes, e.g., GENSCAN (Burge and Karlin, 1997), focus on the prediction of coding exons and exon-intron junctions. Note that introns can be inserted into the 5'-UTR (untranslated region) or 3'-UTR of a gene, creating non-coding exons. Such exons are the most difficult to predict. The first coding exon is the one containing the initiation codon ATG and ending at the first 5' splice junction. The last coding exon is the one containing the termination codon and extending into the 3-UTR. HMMs used in Gene prediction typically involve many hidden states, e.g., GENSCAN involve 17 hidden states.

4. POSTSCRIPT

We see informal applications of HMM in our daily life. By making a telephone call, parents with their ears trained from many years of experience can often detect hidden troubles of their children based only on the voice of the latter. In contrast, people unfamiliar to each other often find it frustratingly difficult to predict each other's behavior. The same applies to different nations, such as between the United States and Iran. If two people or two nations have hardly been trained to understand each other, disasters are almost never far when they both claim to known what the other party intends to do.

I once heard a story about the late Stephen Jay Gould giving a talk on evolution to the congregation of an All Souls Church in New York. When the guest and hosts were having lunch together, someone suggested that they should go around the table to introduce themselves. At that point Gould said something that seemed to be extraordinarily rude, something to the effect that he did not really care who the hosts were as he would never see them again. The name of Gould instantly became synonymous to rudeness among the church members.

However, soon after the incident, the members of the church learned from the newspaper that Gould had died of cancer and that his lecture in the church was in fact Gould's last public engagement – he reserved all the rest of his time to finish his 1464-page magnum opus entitled "The Structure of Evolutionary Theory". They realized that, at that moment when the seemingly rude remark erupted, Gould must have felt melancholy, as everyone would, knowing that his days were numbered, and that he was merely stating a heart-breaking truth that he would never see anyone around the table again.

Stephen Jay Gould had spent all his life fighting two kinds of fundamentalists, the religious fundamentalists who believe that God is a micromanager of everything and the evolutionary fundamentalists who believe that every bit of biodiversity manifests adaptation and every bit of adaptation results from natural selection. I would have expected Gould to have an easy time with members of a very liberal church. Yet misunderstanding still arose, and the misunderstanding could last for along time if Gould's death had not been on the newspaper.

It is truly enigmatic and paradoxical that, with the advanced computational algorithms helping us to infer the unknown, we still do not seem to make any progress in understanding each other and in understanding ourselves. The ancient Greek sage, Plato, has discovered the root cause of all misunderstanding and evil. It is called arrogance. Plato illustrated his point with his famous allegory of the cave.

Imagine prisoners chained inside a cave since childhood, with their heads immobilized in such a way that their eyes were fixed on a gigantic wall. Immediately behind the prisoners was a road along which men, animals and other things traveled. Behind the road was an enormous fire that projected the shadow of the travelers to the wall that the prisoners were facing. Also, the voice of the travelers was echoed from the wall in such a way that the prisoners believed that the words came from the shadows. Gradually, the prisoners became quite good at identifying the travelers by their shadows and voices. The shadows and the voices, as well as the interpretation of the shadows and voices by the prisoners, constituted the reality of the prisoners.

Now suppose a prisoner was freed and went outside the cave. Gradually he would comprehend a new reality from what he could sense. Once thus enlightened, he naturally would want to return to the cave to convey the new reality to his fellow prisoners. Unfortunately, once back in the cave, he found himself much less able to identify the travelers by their shadows than his fellow prisoners. Being thus perceived as inferior by his fellow prisoners, he failed completely in communicating the new reality to his fellow prisoners who believed to know better. The fellow prisoners were too arrogant to listen.

It is the arrogance in the mind of the prisoners that prevents them from comprehending the new reality. It is the arrogance in the mind of the religious fundamentalists and the evolutionary fundamentalists that prevents them from understanding each other. It is the arrogance in the mind of the presidents and primer ministers that prolongs the misunderstanding among nations. An arrogant mind can never perceive the need of training. In the Christian Bible, arrogance is called Satan.

The fundamental message from this chapter is this. An HMM algorithm, no matter how algorithmically elegant and mathematically rigorous, will be of absolutely no value if it is not properly trained. May our mind never be transformed into such an algorithm in reality.

Chapter 7

GIBBS SAMPLER

Identify functional motifs in DNA and proteins

1. INTRODUCTION

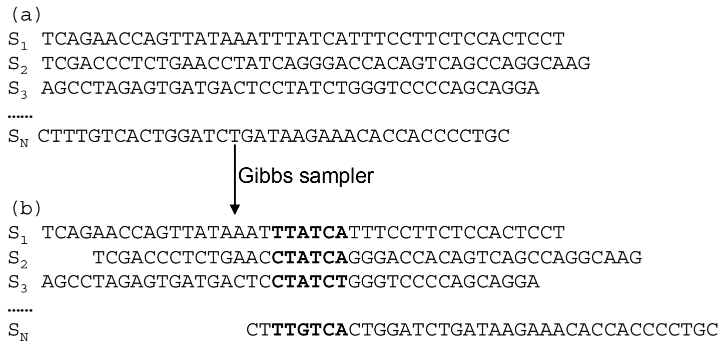
In Chapters 5 on the position weight matrix and perceptron, we have learned how to characterize a sequence motif by using a set of aligned training sequences. This chapter introduces a new computational technique used to identify regulatory motifs in DNA or functional motifs in proteins. There are in fact a number of computational techniques for such identifications. However, the most widely used technique is perhaps Gibbs sampler (Geman and Geman, 1984), named after the mathematical physicist, J. W. Gibbs.

Gibbs sampler is a method for simplifying computation in parameter estimation when analytical solution is very difficult or impossible to obtain. In biology, it has been used in the identification of functional motifs in proteins (Mannella *et al.*, 1996; Neuwald *et al.*, 1995; Qu *et al.*, 1998), biological image processing (Samso *et al.*, 2002), pairwise sequence alignment (Zhu *et al.*, 1998) and multiple sequence alignment (Holmes and Bruno, 2001; Jensen and Hein, 2005). However, the most frequent biological application of Gibbs sampler remains in the identification of regulatory sequences of genes (Aerts *et al.*, 2005; Coessens *et al.*, 2003; Lawrence *et al.*, 1993; Qin *et al.*, 2003; Thijs *et al.*, 2001; Thijs *et al.*, 2002a; Thijs *et al.*, 2002b; Thompson *et al.*, 2004; Thompson *et al.*, 2003).

It is important to recognize the fact that a genome comes alive mainly through transcription and translation. The efficiency of both transcription and translation depends on the associated sequence motifs, with transcription affected by promoter sequences and other associated motifs that can enhance or inhibit transcription, and translation affected by the translation initiation signal such as translation initiation site. These sequence motifs can be discovered by Gibbs sampler.

In this chapter we will focus only on gene and motif prediction involving Gibbs sampler, but the basic algorithm is essentially the same for other applications. Before we detail the algorithm with numerical illustrations, it is helpful to first give a concrete example of its application in motif finding.

Suppose a molecular biologist studying yeast cycle has identified a set of co-expressed genes (i.e., genes that increase or decrease their transcription level synchronously over time) by microarray (Schena, 1996; Schena, 2003) or SAGE (Saha *et al.*, 2002; Velculescu *et al.*, 1995) experiment. He wants to know if the co-expressed genes are also co-regulated, i.e., if they may share a certain yet-unknown promoter sequences controlled by the same or similar transcription factor. Suspecting that the promoter is somewhere upstream of the translation initiation codon, he extracted the upstream sequences from these coexpressed genes (Figure 7-1a) and hope to get the aligned regulatory motifs in the form shown in Figure 7-1b. This scenario is where the Gibbs sampler will shine.



```
(a)
S1 TCAGAACCAGTTATAAATTTATCATTTCCTTCTCCACTCCT
S2 TCGACCCTCTGAACCTATCAGGGACCACAGTCAGCCAGGCAAG
S3 AGCCTAGAGTGATGACTCCTATCTGGGTCCCCAGCAGGA
......
SN CTTTGTCACTGGATCTGATAAGAAACACCACCCCTGC

(b)
S1 TCAGAACCAGTTATAAATTTATCATTTCCTTCTCCACTCCT
S2     TCGACCCTCTGAACCTATCAGGGACCACAGTCAGCCAGGCAAG
S3 AGCCTAGAGTGATGACTCCTATCTGGGTCCCCAGCAGGA
......
SN                 CTTTGTCACTGGATCTGATAAGAAACACCACCCCTGC
```

Figure 7-1. Application of Gibbs sampler in motif discovery. The sequences shown are reverse complement of a subset of erythroid-specific gene sequences, which has been tested for the presence of GATA box or TATC box in reverse complement (Rouchka, 1997).

The main output of Gibbs sampler is typically of two parts. The first is the sequences with aligned motifs as shown in Figure 7-1b. The second is a position weight matrix derived from the aligned motifs so that we can use it to scan new sequences for the presence and location of such motifs.

In short, the input to Gibbs sampler for motif prediction is a set of sequences, the majority of which contain one or more motifs of interest (Figure 7-1a). The output is a set of sequences with aligned motifs (Figure 7-2) together with a position weight matrix that can be used in future motif prediction.

There are two slightly different applications of Gibbs sampler in motif prediction. The first assumes that each sequence contains exactly one motif (Lawrence *et al.*, 1993) and the associated algorithm is called site sampler. The second is more flexible and allows each sequence to have none or multiple motifs (Neuwald *et al.*, 1995) and the algorithm is termed motif sampler. We will illustrate the site sampler and then briefly discuss the motif sampler. Much of this chapter is based on a previous tutorial on Gibbs sampler (Rouchka, 1997).

2. A NUMERICAL ILLUSTRATION OF THE COMPUTATIONAL DETAILS OF GIBBS SAMPLER

We have to first of all face the necessary evil of defining relevant entities. For illustration, we will work with nucleotide sequences instead of amino acid sequences, although the algorithm is applicable to both. The erythroid sequences (Rouchka, 1997) that we will apply Gibbs sampler to are listed below (with 3'-end trimmed to the maximum length 50 bases to fit the screen):

S_1	TCAGAACCAGTTATAAATTTATCATTTCCTTCTCCACTCCT
S_2	CCCACGCAGCCGCCCTCCTCCCCGGTCACTGACTGGTCCTG
S_3	TCGACCCTCTGAACCTATCAGGGACCACAGTCAGCCAGGCAAG
S_4	AAAACACTTGAGGGAGCAGATAACTGGGCCAACCATGACTC
S_5	GGGTGAATGGTACTGCTGATTACAACCTCTGGTGCTGC
S_6	AGCCTAGAGTGATGACTCCTATCTGGGTCCCCAGCAGGA
S_7	GCCTCAGGATCCAGCACACATTATCACAAACTTAGTGTCCA
S_8	CATTATCACAAACTTAGTGTCCATCCATCACTGCTGACCCT
S_9	TCGGAACAAGGCAAAGGCTATAAAAAAAATTAAGCAGC
S_{10}	GCCCCTTCCCCACACTATCTCAATGCAAATATCTGTCTGAAACGGTTCC
S_{11}	CATGCCCTCAAGTGTGCAGATTGGTCACAGCATTTCAAGG
S_{12}	GATTGGTCACAGCATTTCAAGGGAGAGACCTCATTGTAAG
S_{13}	TCCCCAACTCCCAACTGACCTTATCTGTGGGGGAGGCTTTTGA
S_{14}	CCTTATCTGTGGGGGAGGCTTTTGAAAAGTAATTAGGTTTAGC
S_{15}	ATTATTTTCCTTATCAGAAGCAGAGAGACAAGCCATTTCTCTTTCCTCCC
S_{16}	AGGCTATAAAAAAAATTAAGCAGCAGTATCCTCTTGGGGGCCCCTTC
S_{17}	CCAGCACACACACTTATCCAGTGGTAAATACACATCAT
S_{18}	TCAAATAGGTACGGATAAGTAGATATTGAAGTAAGGAT
S_{19}	ACTTGGGGTTCCAGTTTGATAAGAAAAGACTTCCTGTGGA
S_{20}	TGGCCGCAGGAAGGTGGGCCTGGAAGATAACAGCTAGTAGGCTAAGGCCA
S_{21}	CAACCACAACCTCTGTATCCGGTAGTGGCAGATGGAAA

```
S22 CTGTATCCGGTAGTGGCAGATGGAAAGAGAAACGGTTAGAA
S23 GAAAAAAAATAAATGAAGTCTGCCTATCTCCGGGCCAGAGCCCCT
S24 TGCCTTGTCTGTTGTAGATAATGAATCTATCCTCCAGTGACT
S25 GGCCAGGCTGATGGGCCTTATCTCTTTACCCACCTGGCTGT
S26 CAACAGCAGGTCCTACTATCGCCTCCCTCTAGTCTCTG
S27 CCAACCGTTAATGCTAGAGTTATCACTTTCTGTTATCAAGTGGCTTCAGC
S28 GGGAGGGTGGGGCCCCTATCTCTCCTAGACTCTGTG
S29 CTTTGTCACTGGATCTGATAAGAAACACCACCCCTGC
```

Let N be the number of input sequences designated as S_1, S_2, …, S_i, …, S_N, and m be the length of the motif. For our example, N = 29 and m = 6. One typically would use different m values if one knows little about the length of the motif.

Let L_i be the length of S_i, and A_i be the inferred starting position of the motif in S_i. The objective of Gibbs sampler is to (1) obtain a set of correct A_i values to align the motifs in the form of Figure 7-1b, and (2) generate a position weight matrix that characterizes the motif by site-specific nucleotide frequency distributions. The position weight matrix can be used to scan for the presence of the identified motif for purpose of motif prediction. The position weight matrix is of dimension m×4, for nucleotide sequences and m×20, for amino acid sequences.

We need first to count all nucleotides, with their numbers designated as F_A, F_C, F_G and F_T, respectively, in the sequences. The total number of nucleotides of all 29 sequences is 1209, with F_A, F_C, F_G and F_T equal to 325, 316, 267 and 301, respectively. These values will be needed for calculating pseudocounts (which we encountered in the section on position weight matrix in Chapter 5).

The main algorithm of Gibbs sampler is of two steps. The first is random initialization in which a random set of A_i value is chosen and site-specific nucleotide frequencies are calculated. The second step is predictive updating until a local solution of A_i values is obtained and retained, together with site-specific nucleotide frequencies that can be made into a position weight matrix. This is repeated multiple times and previously stored locally optimal solutions are replaced by better ones. Convergence is typically declared when two or more local solutions are identical. These steps are numerically illustrated in the following sections.

2.1 Initialization

We now randomly assign a value to A_i, with the constraint that $A_i \le L_i - m + 1$. So our first set of N "motifs" is essentially a random set of sequences of length m and is not expected to have any pattern. Just in case you are curious, the first set of 29 random A_i values happen to be: 29, 31, 23, 28, 10, 2, 18, 32, 20, 15, 11, 25, 24, 30, 18, 15, 10, 23, 14, 15, 26, 36, 8, 6, 30, 19, 27, 26, and 14. The site-specific distribution of nucleotides from the 29 random motifs is shown in Table 7-1. There is hardly any site-specific pattern.

The second column in Table 7-1 will be referred to as C0 vector with $C0_A$, $C0_C$, $C0_G$ and $C0_T$ equal to 278, 279, 230, and 248, respectively. The 4×6 matrix, occupying the last six columns in Table 7-1, will be referred to as C matrix. The C matrix is tabulated from the 29 random motifs whereas the C0 vector is tabulated from nucleotides outside of the motifs. Thus, the sum of the first, second, third and fourth rows should be equal to F_A, F_C, F_G and F_T, respectively. Also note that each of the six columns in the C matrix should add up to 29.

Table 7-1. Site-specific distribution of nucleotides from the 29 random motifs of length 6. The second column lists the distribution of nucleotides outside the 29 random motifs.

		Site					
Nuc	C0	1	2	3	4	5	6
A	278	8	7	9	6	10	7
C	279	3	8	5	10	6	5
G	230	7	5	6	5	3	11
T	248	11	9	9	8	10	6

2.2 Predictive update

The predictive update consists of obtaining N (= 29 in our example) random numbers ranging from 1 to N, and use these numbers as an index to choose the sequences sequentially to update the site-specific distribution of nucleotides (the C matrix) and the associated frequencies (the C0 vector). For example, the N numbers in my run of the Gibbs sampler happen to be 11, 18, 26, 22, 2, 28, 12, 9, 7, 3, 17, 16, 1, 4, 21, 15, 14, 24, 19, 27, 29, 6, 10, 20, 13, 8, 23, 25, and 5, respectively. This means that S_{11} will be used first, and S_5 last, for the first cycle of the predictive update. It is important to use a random series of numbers instead of choosing sequences according to the input order. The latter increases the likelihood of trapping Gibbs sampler within a local optimum. This is repeated multiple times until a local solution is reached. I present the detail of the predictive update below.

Our first randomly chosen sequence happens to be S_{11} and its randomly chosen motif, as has mentioned in the previous section, happens to start at 11, i.e., A_{11} = 11, with the motif being AGTGTG. This initial motif will now be taken out of C and put into C0 vector. This motif has one A, zero C, and three G's and two U's. By adding these values to the C0 vector in Table 7-1, we obtain the C0 vector in Table 7-2. We also need to take this motif out of the C matrix by subtracting the first A from the first value in the first column in the C matrix in Table 7-1 (i.e., new $C_{A,1}$ = old $C_{A,1}$ - 1) , the second G from the third value in the second column in the C matrix in Table 7-1 (i.e., new $C_{G,2}$ = old $C_{G,2}$ - 1), and so on. This converts the C matrix in Table 7-1 to the C matrix in Table 7-2.

Table 7-2. Site-specific distribution of nucleotides from the 28 random motifs of length 6, after removing the initial motif in S_{11}. The second column lists the distribution of nucleotides outside the 28 random motifs.

		Site					
Nuc	C0	1	2	3	4	5	6
A	279	7	7	9	6	10	7
C	279	3	8	5	10	6	5
G	233	7	4	6	4	3	10
T	250	11	9	8	8	9	6

At this point the C matrix is made of the 28 randomly chosen motifs, one from each sequence. You will notice that each of the six columns in the C matrix has a sum of 28.

The reason for taking the initial motif in S_{11} out of the C matrix and put it back into the C0 vector is that we are going to find a more likely motif in S_{11}, and put it into the C matrix so that the C matrix will against be based on 29 motifs. How are we going to get a more likely motif? Recall that a position weight matrix (PWM) can be used to scan fragments of a sequence to get position weight matrix scores (PWMSs). We will make a PWM out of the C0 vector and the C matrix and use the resulting PWM to scan S_{11} and get a new motif that has the highest PWMS.

You may wonder why such a practice would get us anywhere given the fact that the C matrix is initially made of random motifs. The resulting PWM would exhibit no pattern, and the resulting PWMSs will therefore be uninformative. It is a valid point that you have made, but let us wait and see if some miracles will happen.

With the C0 vector and C matrix in Table 7-2, we will now create a new Q0 vector and a new Q matrix. The four values in the Q0 vector are computed as

$$Q0_i = \frac{C0_i}{\sum_{i=1}^{N_{Code}} C0_i}, e.g.,$$
$$Q0_A = \frac{279}{279+279+233+250} = 0.268012 \tag{7.1}$$

where N_{Code} is the number of different symbols in the sequences (= 4 for nucleotide sequences, and i = 1, 2, 3 and 4 corresponding to A, C, G and T).

Thus, the Q0 vector is just the proportion of A, C, G and T in the 28 sequences outside of the 28 motifs. However, because $C0_i$ may be zero, and because we will need to take the logarithm of $Q0_i$, we will add pseudocounts to obtain $Q0_i$ in the following form.

$$Q0_i = \frac{C0_i + \alpha F_i}{\sum_{i=1}^{N_{Code}} C0_i + \alpha \sum_{i=1}^{N_{Code}} F_i} \tag{7.2}$$

where F_i has been defined before , and α is typically a small value, with its default being 0.01 in my implementation of Gibbs Sampler in DAMBE (Xia, 2001; Xia and Xie, 2001b).

The elements in the Q matrix are computed similarly as follows:

$$Q_{ij} = \frac{C_{ij} + \alpha F_i}{\sum_{i=1}^{N_{Code}} C_{ij} + \alpha \sum_{i=1}^{N_{Code}} F_i} = \frac{C_{ij} + \alpha F_i}{N - 1 + \alpha \sum_{i=1}^{N_{Code}} F_i} \tag{7.3}$$

where i = 1, 2, 3, and 4 corresponding to A, C, G and T, respectively, N is the number of input sequences, and j = 1 2, …, m. The Q0 vector and the Q matrix are shown in Table 7-3.

Table 7-3. Site-specific distribution of nucleotide frequencies derived from data in Table 7-2. The second column lists the distribution of nucleotide frequencies outside the 28 random motifs.

		Site					
Nuc	Q0	1	2	3	4	5	6
A	0.2680	0.2495	0.2495	0.2981	0.2251	0.3225	0.2495
C	0.2679	0.1499	0.2716	0.1986	0.3203	0.2229	0.1986
G	0.2238	0.2353	0.1623	0.2110	0.1623	0.1380	0.3083
T	0.2403	0.3410	0.2923	0.2679	0.2679	0.2923	0.2193

Readers who have forgotten PWM might benefit from reviewing Chapter 5 on PWM. Recall that PWM can be used to obtain a PWM score (PWMS) for a motif of length m and the PWMS value measures our confidence in one hypothesis (the motif shares the site dependence as those 28 "motifs" contributing to the C matrix, designated as θ_{Yes}) relative to its alternative (θ_{No}), given a motif.

With the Q0 vector and the Q matrix in Table 7-3, we can now scan S_{11} for a more likely motif. We compute a PWMS for each motif stating point.

For example, for starting point at 1, we have the first 6-mer from S_{11} equal to CATGCC. I provide you with computational details just in case you do not feel like to review Chapter 5 on PWM:

$$\begin{aligned} L_{Yes} &= p(S \mid \theta_{Yes}) = Q_{C,1}Q_{A,2}Q_{U,3}Q_{G,4}Q_{C,5}Q_{C,6} = 0.000072 \\ L_{No} &= p(S \mid \theta_{No}) = Q0_A Q0_C^3 Q0_G Q0_T = 0.000277 \\ PWMS &= \frac{L_{Yes}}{L_{No}} = 0.259787 \end{aligned} \tag{7.4}$$

In actual computation, we generally would have taken logarithms to avoid possible computer overflow or underflow errors that often occur when the motif is long.

S_{11} is 40 bases long, with 35 (= 40 – m + 1) possible motif starting points (i.e., possible A_i values along the sequence). The 35 PWMS values for these 35 possible motifs (Table 7-4) are normalized to have a sum of 1 (P_{Norm} in Table 7-4). We now proceed to update the initial A_{11} (=11) by a new A_{11} value based on result in Table 7-4. How should we choose the new A_{11} value?

There are two strategies to choose the new A_{11} value. The first is to randomly pickup an A_i value according to the magnitude of P_{Norm} (Table 7-4). You may visualize a dartboard with 35 slices with their respective areas being proportional to P_{Norm} values. When you throw a dart at the dartboard, large slices will have a better chance of being hit than small slices. If the dart happens to land on the 7th slice, then the initial A_{11} = 11 will be updated to A_{11} = 7, with the original motif AGTGTG replaced by the new motif CTCAAG.

The second strategy is simply to use the largest P_{Norm} value for updating initial A_{11} to the new A_{11} value. As the motif starting at site 25 has the largest P_{Norm}, we will set the new A_{11} equal to 25 and replace the initial motif (=AGTGTG) by the new motif (=TCACAG). This strategy is faster than the first, but seems to be just as sensitive in motif detection as the first. My own limited computer simulation does not seem to indicate that the second strategy is more likely to trap Gibbs sampler in a local optimum. It may sound odd, but there has been no systematic studies evaluating the effectiveness of these two strategies.

Regardless of how the new A_{11} is chosen, the updating is the same. Suppose we have taken the second strategy and set the new A_{11} equal to 25. The C matrix in Table 7-1 is then revised by replacing the original A_{11} motif (=AGTGTG) by the new motif (=TCACAG). This leads to an updated C0 vector and C matrix (Table 7-5).

Table 7-4. Possible locations of the 6-mer motif along S_{11}, together with the corresponding motifs and their position weight matrix scores expressed as odds ratios. The last column lists the odds ratios normalized to have a sum of 1.

Site	6-mer	Odds Ratio	P_{Norm}
1	CATGCC	0.2598	0.0079
2	ATGCCC	0.7869	0.0240
3	TGCCCT	0.6925	0.0211
4	GCCCTC	0.8516	0.0259
5	CCCTCA	0.3630	0.0111
6	CCTCAA	0.8467	0.0258
7	CTCAAG	0.7025	0.0214
8	TCAAGT	0.7563	0.0230
9	CAAGTG	0.7043	0.0214
10	AAGTGT	0.5126	0.0156
11	AGTGTG	0.9155	0.0279
12	GTGTGC	0.6148	0.0187
13	TGTGCA	0.6449	0.0196
14	GTGCAG	2.3902	0.0728
15	TGCAGA	0.3678	0.0112
16	GCAGAT	0.9444	0.0287
17	CAGATT	0.4579	0.0139
18	AGATTG	1.4038	0.0427
19	GATTGG	1.0343	0.0315
20	ATTGGT	0.5155	0.0157
21	TTGGTC	1.0647	0.0324
22	TGGTCA	0.8382	0.0255
23	GGTCAC	0.9068	0.0276
24	GTCACA	0.6167	0.0188
25	TCACAG	3.1708	0.0965
26	CACAGC	0.1482	0.0045
27	ACAGCA	0.5895	0.0179
28	CAGCAT	0.6445	0.0196
29	AGCATT	0.4666	0.0142
30	GCATTT	1.4683	0.0447
31	CATTTC	0.5841	0.0178
32	ATTTCA	1.0906	0.0332
33	TTTCAA	2.5773	0.0784
34	TTCAAG	1.7817	0.0542
35	TCAAGG	1.1418	0.0348

Table 7-5. Site-specific distribution of nucleotides from the 29 initial motifs of length 6, after replacing the initial A_{11} motif (=AGTGTG) by the new motif (=TCACAG).

		Site					
Nuc	C0	1	2	3	4	5	6
A	277	7	7	10	6	11	7
C	277	3	9	5	11	6	5
G	232	7	4	6	4	3	11
T	249	12	9	8	8	9	6

We repeat this process for the rest of the sequences to update the rest of A_i values. After the last sequence has been updated, we have obtained a new set of A_i values, a new set of 29 motifs, together with the associated C matrix and Q matrix based on the site-specific frequencies of these motifs. At this point we compute a weighted alignment score (i.e., a weighted PWMS) as follows:

$$F = \sum_{i=1}^{N_{Code}} \sum_{j=1}^{m} C_{i,j} \ln \frac{Q_{ij}}{Q0_i} \tag{7.5}$$

where m is the motif width, and N_{Code} is the number of different symbols in the sequences (4 for nucleotide and 20 for amino acid sequences). F is a measure of the quality of alignment of the motifs. The larger the F value, the better.

The predictive updating is repeated again and again. Each time when we get a new set of A_i values, a new set of motifs and the associated Q0 vector and Q matrix, we compute a new F value. If the new F value is greater than the previously stored F value, then new F value, the new set A_i values, and the new set of motifs will replace the previously stored ones. This continues until we reach a local maximum of F or when the preset maximum number of local loops has been reached. The resulting F value, the set of A_i values, the new set of motifs and the associated PWM are stored as the locally optimal output.

This process is repeated from the very beginning, i.e., we again perform the initialization by choosing a random set of A_i values, and go through the local iteration to obtain another locally optimal output. If the new locally optimal output is better than previously stored ones (i.e., the new F value is larger than the previously stored one), the new output will replace the previously stored output. This process is repeated multiple times until convergence is reached, i.e., when new F values are the same as the previously stored one. The final site-specific nucleotide distribution (Table 7-6) displays a much stronger pattern than the initial distribution (Table 7-1) from 29 randomly chosen motifs.

Table 7-6. Final site-specific distribution of nucleotides from the 29 identified motifs. Output from DAMBE (Xia, 2001; Xia and Xie, 2001b).

		Site					
Nuc	C0	1	2	3	4	5	6
A	275	3	0	22	0	9	16
C	285	11	0	0	0	19	1
G	252	0	7	7	0	0	1
T	223	15	22	0	29	1	11

The final aligned motifs (Figure 7-2) share in general a consensus of (C/T)TATC(A/T). Its reverse complement (A/T)GATA(A/G) is known to be the binding site of GATA-binding transcription factors (Aird *et al.*, 1994; Fong and Emerson, 1992; Moi *et al.*, 1992; Nishimura *et al.*, 2000; Orkin, 1992; Zon *et al.*, 1991). This discovery of the motif suggests that this set of sequences may indeed be co-regulated by the same type of GATA-binding transcription factors. Such findings are crucial in transcriptomic and proteomic studies aiming to understand gene regulation networks. Although we are already in the so-called post-genomic era, we actually know little about how genomes work. What we have is mostly a list of genes. Algorithms such as Gibbs sampler help us understand interactions among genes and gene products.

```
 1               TCAGAACCAGTTATAAATTTATCATTTCCTTCTCCACTCCT
 2           CCCACGCAGCCGCCCTCCTCCCCGGTCACTGACTGGTCCTG
 3                   TCGACCCTCTGAACCTATCAGGGACCACAGTCAGCCAGGCAAG
 4                AAAACACTTGAGGGAGCAGATAACTGGGCCAACCATGACTC
 5                 GGGTGAATGGTACTGCTGATTACAACCTCTGGTGCTGC
 6               AGCCTAGAGTGATGACTCCTATCTGGGTCCCCAGCAGGA
 7             GCCTCAGGATCCAGCACACATTATCACAAACTTAGTGTCCA
 8                               CATTATCACAAACTTAGTGTCCATCCATCACTGCTGACCCT
 9                TCGGAACAAGGCAAAGGCTATAAAAAAAATTAAGCAGC
10                   GCCCCTTCCCCACACTATCTCAATGCAAATATCTGTCTGAAACGGTTCC
11            CATGCCCTCAAGTGTGCAGATTGGTCACAGCATTTCAAGG
12GATTGGTCACAGCATTTCAAGGGAGAGACCTCATTGTAAG
13             TCCCCAACTCCCAACTGACCTTATCTGTGGGGGAGGCTTTTGA
14                               CCTTATCTGTGGGGGAGGCTTTTGAAAAGTAATTAGGTTTAGC
15                       ATTATTTTCCTTATCAGAAGCAGAGAGACAAGCCATTTCTCTTTCCTCCC
16                              AGGCTATAAAAAAAATTAAGCAGCAGTATCCTCTTGGGGGCCCCTTC
17                    CCAGCACACACACTTATCCAGTGGTAAATACACATCAT
18             TCAAATAGGTACGGATAAGTAGATATTGAAGTAAGGAT
19                 ACTTGGGGTTCCAGTTTGATAAGAAAAGACTTCCTGTGGA
20         TGGCCGCAGGAAGGTGGGCCTGGAAGATAACAGCTAGTAGGCTAAGGCCA
21                     CAACCACAACCTCTGTATCCGGTAGTGGCAGATGGAAA
22                                 CTGTATCCGGTAGTGGCAGATGGAAAGAGAAACGGTTAGAA
23          GAAAAAAAATAAATGAAGTCTGCCTATCTCCGGGCCAGAGCCCCT
24                             TGCCTTGTCTGTTGTAGATAATGAATCTATCCTCCAGTGACT
25                GGCCAGGCTGATGGGCCTTATCTCTTTACCCACCTGGCTGT
26                  CAACAGCAGGTCCTACTATCGCCTCCCTCTAGTCTCTG
27              CCAACCGTTAATGCTAGAGTTATCACTTTCTGTTATCAAGTGGCTTCAGC
28                  GGGAGGGTGGGGCCCCTATCTCTCCTAGACTCTGTG
29                               CTTTGTCACTGGATCTGATAAGAAACACCACCCCTGC
```

Figure 7-2. Aligned motifs generated from Gibbs sampler. Output from DAMBE (Xia, 2001; Xia and Xie, 2001b).

I consider it relevant to provide a summary of the GATA box and GATA-binding transcription factors so that you can better appreciate the application of Gibbs sampler in the proper biological context. A living cell is a system with many genetic switches that can be turned on or off in response to intracellular and extracellular environment. It is these switches that distinguish a normal living cell from a cancer cell or a dead cell. The GATA motif (or GATA box) is one of such switches and it is switched on by specific transcription factors (which are proteins that bind to the motif and turn on or off the transcription of the gene containing such motifs). One of the better known GATA-binding transcription factors is GATA-1 which binds to the GATA motif found in cis-elements of the vast majority of erythroid-expressed genes of all vertebrate species examine (Evans *et al.*, 1990; Orkin, 1990). The core promoter of the rat platelet factor 4 (PF4) gene contains such a GATA motif and the binding of such GATA motif by GATA-binding proteins such as GATA-1 suppresses the transcription of the PF4 gene (Aird *et al.*, 1994). It is now known that GATA regulatory motifs and the GATA-binding transcription factors are present in a variety of organisms ranging from cellular slime mold to vertebrates, including plants, fungi, nematodes, insects, and echinoderms (Lowry and Atchley, 2000), suggesting that the function of the genetic switch is far beyond erythropoiesis. In human, the GATA motif and the GATA-binding proteins are implicated in several diseases (Van Esch and Devriendt, 2001).

You may have noted that some sequences have a strong (C/T)TATC(A/T) motif, whereas others (e.g., the second, the fourth and the fifth sequences) have only weak and highly doubtful signals. Computer programs implementing Gibbs sampler typically would output a quantitative measure of the strength of the signal, and PWMS is the most often used index for this purpose (Table 7-7). Recall that PWMS in Chapter 5 is the log-odds, but here we use the odds ratio directly as a measure of motif strength. Also recall that an odds ratio is the ratio of two probabilities associated with two hypotheses. Define θ_{Yes} as the hypothesis that the 6-mer is a motif with its probability specified by the position weight matrix derived form the site-specific nucleotide frequencies in Table 7-5, and θ_{No} as the hypothesis that the 6-mer is not a motif and has its probabilities specified only by the four overall nucleotide frequencies. The odds ratio is the ratio of the probability that θ_{Yes} is true over the probability that θ_{No} is true. One generally should take a cutoff value of 20, i.e., θ_{Yes} is 20 times more likely than θ_{No}. In other words, the chance that θ_{Yes} is true is 95%.

From a statistical estimation point of view, the matrix in Table 7-5, or the position weight matrix that can be derived from it, is in fact what we wish to obtain. This matrix is the result of training the Gibbs sampler with the 29 sequences in the training set, and it allows us to go beyond the motifs in the training sequences to scan any unknown sequences or an entire genome to find similar motifs.

Table 7-7. Output of PWMS as a quantitative measure of the strength of the identified motifs. Output from DAMBE (Xia, 2001; Xia and Xie, 2001b).

SeqName	Motif	Start	Odds-ratio
Seq1	TTATCA	18	163.6602
Seq2	CGGTCA	22	14.5511
Seq3	CTATCA	14	101.8203
Seq4	AGATAA	17	9.1127
Seq5	TGATTA	16	12.9266
Seq6	CTATCT	18	90.7790
Seq7	TTATCA	20	163.6602
Seq8	TTATCA	2	163.6602
Seq9	CTATAA	17	58.1420
Seq10	CTATCT	14	90.7790
Seq11	TGGTCA	21	23.3886
Seq12	TTGTAA	33	38.9024
Seq13	TTATCT	20	145.9129
Seq14	TTATCT	2	145.9129
Seq15	TTATCA	10	163.6602
Seq16	CTATAA	3	58.1420
Seq17	TTATCC	13	34.3258
Seq18	AGATAT	20	8.1245
Seq19	TGATAA	16	32.0835
Seq20	AGATAA	24	9.1127
Seq21	CTGTAT	12	21.5783
Seq22	CTGTAT	0	21.5783
Seq23	CTATCT	23	90.7790
Seq24	TTGTCT	4	60.7395
Seq25	TTATCT	17	145.9129
Seq26	CTATCG	15	21.2368
Seq27	TTATCA	19	163.6602
Seq28	CTATCT	15	90.7790
Seq29	TTGTCA	2	68.1272
Mean			76.3120
Stdev			57.8163

I should emphasize the fact here that Gibbs sampler, being started from random motif selection, may not necessarily converge to the same motif. This is both an advantage and a disadvantage of the algorithm. The advantage is that repeated running of the algorithm will allow us to identify other types of hidden motifs (i.e., other than the reverse complement of the GATA motif) in the sequences. The disadvantage is that users not familiar with the algorithm often get confused when the same input generates quite different results. For example, another set of putative motifs is partially shown in Figure 7-3.

```
 4                 AAAACACTTGAGGGAGCAGATA...
12           GATTGGTCACAGCATTTCAAGGGAGAGA...
13 TCCCCAACTCCCAACTGACCTTATCTGTGGGGGAGGCT...
14                   CCTTATCTGTGGGGGAGGCT...
15             ATTATTTTCCTTATCAGAAGCAGAGA...
20          TGGCCGCAGGAAGGTGGGCCTGGAAGATA...
21     CAACCACAACCTCTGTATCCGGTAGTGGCAGATG...
22                 CTGTATCCGGTAGTGGCAGATG...
26                           CAACAGCAGGTC...
28                               GGGAGGGT...
```

Figure 7-3. An alternative motif that may be returned from running the Gibbs sampler on the same set of input sequences. The left column lists the sequence numbers.

3. MOTIF SAMPLER

The Gibbs sampler has two versions. The one that we have just illustrated is called site sampler. It assumes that each sequence contains exactly one motif (Lawrence *et al.*, 1993). The other version is more flexible and allows each sequence to have none or multiple motifs (Neuwald *et al.*, 1995) and the algorithm is termed motif sampler. The GATA-binding transcription factors comprise a protein family whose members contain either one or two highly conserved zinc finger DNA-binding domains (Lowry and Atchley, 2000) and it is consequently likely that a sequence may contain more than one GATA box. For example, the erythroid Kruppel-like factor (EKLF, which is a zinc finger transcription factor required for β-globin gene expression) has in its 5'-region two GATA motifs flanking an E box motif characterized by CANNTG (Anderson *et al.*, 1998). This calls for an algorithm that can identify multiple motifs in a single sequence.

The site sampler can be extended to motif sampler by post-processing. The C matrix and C0 vector in Table 7-5 can be made into a position weight matrix to re-scan the sequences for motifs and compute the associated PWMS or odds ratio for all 6-mers in each sequence. All what we need is to have a cutoff score to keep those motifs with a PWMS or odds ratio greater than the cutoff score. An odds ratio of 20 is a reasonable cutoff score. Table 7-7 shows the output of motif sampler with a cutoff score of 10.

Table 7-8. Motif sampler output from DAMBE (Xia, 2001; Xia and Xie, 2001b). N: number of motifs in the sequence; columns under headings 1, 2 and 3 lists details of the identified motif in the format of "Start(Motif, Odds ratio)"

SeqName	N	1	2	3
Seq1	2	10(TTATAA,93.4541)	18(TTATCA,163.6602)	
Seq2	1	22(CGGTCA,14.5511)		
Seq3	1	14(CTATCA,101.8203)		
Seq4	0			
Seq5	1	16(TGATTA,12.9266)		
Seq6	1	18(CTATCT,90.7790)		
Seq7	1	20(TTATCA,163.6602)		
Seq8	2	2(TTATCA,163.6602)	24(CCATCA,10.2098)	
Seq9	1	17(CTATAA,58.1420)		
Seq10	3	14(CTATCT,90.7790)	28(ATATCT,41.4438)	32(CTGTCT,37.7888)
Seq11	1	21(TGGTCA,23.3886)		
Seq12	2	3(TGGTCA,23.3886)	33(TTGTAA,38.9024)	
Seq13	1	20(TTATCT,145.9129)		
Seq14	1	2(TTATCT,145.9129)		
Seq15	3	1(TTATTT,33.5700)	10(TTATCA,163.6602)	36(TTCTCT,17.7407)
Seq16	1	3(CTATAA,58.1420)		
Seq17	2	13(TTATCC,34.3258)	21(TGGTAA,13.3555)	
Seq18	0			
Seq19	1	16(TGATAA,32.0835)		
Seq20	0			
Seq21	1	12(CTGTAT,21.5783)		
Seq22	1	0(CTGTAT,21.5783)		
Seq23	1	23(CTATCT,90.7790)		
Seq24	2	4(TTGTCT,60.7395)	26(CTATCC,21.3556)	
Seq25	1	17(TTATCT,145.9129)		
Seq26	1	15(CTATCG,21.2368)		
Seq27	3	19(TTATCA,163.6602)	25(CTTTCT,13.3635)	32(TTATCA,163.6602)
Seq28	1	15(CTATCT,90.7790)		
Seq29	2	2(UUGUCA,68.1272)	15(TGATAA,32.0835)	

Chapter 8

BIOINFORMATICS AND VERTEBRATE MITOCHONDRIA

A glimse into the three essential biological processes

1. INTRODUCTION

This chapter has two main purposes. First, I believe that we again need to add some variation to the monotonous presentation of computational algorithms. The relationship between bioinformatics and the cell is analogous to that between hair and skin. Without the cell serving as the skin the hair of bioinformatics would have no place to grow. So here we take a close look at the skin, in the same spirit as in chapter 4. If you happen to be the colleague who told me that Chapter 4 was a tumor on an otherwise beautiful nymph, I am afraid that another tumor is coming your way.

The second purpose of the chapter is to demonstrate the utility of a well annotated genome. We have learned much about how to assemble a genome and how to use gene and motif finding methods to annotate the genome. It is natural for one to ask why we should spend so much effort to obtain an annotated genome. This chapter provides a partial answer, i.e., well annotated genomes can be used to test biologically far more interesting hypotheses than those we encountered in Chapter 4, where our analysis is limited to genomic GC% without taking advantage of sequence annotation.

I have mentioned before that all living systems share three essential biological processes: genome replication, transcription and translation. How cells function depends on how these three processes are regulated. In this chapter we take a quick look at these three processes, not in the context of a living cell, but in a much simpler system, the vertebrate mitochondrion.

1.1 Mitochondria and mitochondrial genomes

Since the proposal of the endosymbiotic origin of eukaryotic mitochondria (Margulis, 1970), it has now been generally accepted that mitochondria of eukaryotes evolved from aerobic bacteria living within their host cell, with the most likely ancestor being a member in the Rickettsia lineage. There are three lines of evidence supporting this endosymbiotic hypothesis. First, genomic analysis of the eukaryotic intracellular parasite, *Rickettsia prowazekii*, has reveal that the functional profiles of *R. prowazekii* genes are similar to those of mitochondrial genes and phylogenetic reconstruction indicates that *R. prowazekii* is more closely related to mitochondria than is any other microbe studied (Andersson *et al.*, 1998). Second, genomes from various *Rickettsia* species have exhibited various degrees of degradation and reduction (Andersson and Andersson, 1999), suggesting that mitochondria represent extremely reduced forms from certain ancestral *Rickettsia* lineage. Third, mitochondrial genomes from primitive protozoans such as *Reclinomonas americana* containing genes specifying a multi-subunit eubacterial-type RNA polymerase.

There are several eukaryotic lineages that do not have mitocondria. Among these amitochondriate lineages, the retortamonads is most likely to represent the ancestral amitochondriate state (Lang *et al.*, 1997), whereas others such as *Encephalitozoon cuniculi* may result from a secondary loss of the organelle (Katinka *et al.*, 2001). Thus, the mitochondrial eukaryotic lineage that is the closest relative of the retortamonads likely represents the oldest lineage where ancestral mitochondria originated.

The closest relative to the amitochondriate retortamonads is the Jakobid assemblage represented by *Reclinomonas americana*, a heterotrophic flagellate, with a mitochondrial genome much larger and more complicated than the one found in vertebrate mitochondrion (Lang *et al.*, 1997). The *R. americana* mtDNA is of 69034 bases long, contains 97 genes with 4 genes specifying a multi-subunit eubacterial-type RNA polymerase. In contrast, vertebrate mitochondrial genomes are about 16500 bases long, and contain only 13 protein-coding genes, 2 rRNA genes, and about 22 tRNA genes.

It is quite obvious that genomes evolve over time and can experience dramatic changes. From an evolutionary point of view, there are only two major sculptors of nature, mutation and selection. Of course there are other factors contributing to evolutionary changes but we will limit ourselves to only mutation and selection. Anyone who, in their young days, has ever involved in a love triangle would know how complex and unmanageable our life could become when we go from two to three.

Here we will learn a few important observations on vertebrate mitochondrial replication and illustrate how these two factors interact with the three essential biological processes, i.e., DNA replication, transcription and translation. What is particularly relevant to bioinformaticians is that such interactions will generate signals that can be detected, characterized and studied.

1.2 DNA-Replication and Strand-biased mutation spectrum

Vertebrate mitochondrial genome has two strands of different buoyant densities and consequently named the H-strand and the L-strand. The H-strand is the sense strand for one protein-coding gene (ND6) and 8 tRNA genes and the L-strand is the sense strand for 12 protein-coding genes, 2 rRNA genes and 14 tRNA genes. The two strands have different nucleotide frequencies, with the H-strand rich in G and T and the L-strand rich in A and C (Jermiin *et al.*, 1995; Perna and Kocher, 1995). This asymmetrical distribution of nucleotides has been explained as follows (Reyes *et al.*, 1998; Tanaka and Ozawa, 1994) based on the strand-displacement model of mitochondrial DNA (mtDNA) replication (Bogenhagen and Clayton, 2003; Clayton, 1982, 2000; Shadel and Clayton, 1997).

During mtDNA replication (Figure 8-1), the L-strand is first used as a template to replicate the daughter H-strand, while the parental H-strand was left single-stranded for an extended period because the complete replication of vertebrate mtDNA takes nearly two hours (Clayton, 1982, 2000; Shadel and Clayton, 1997).

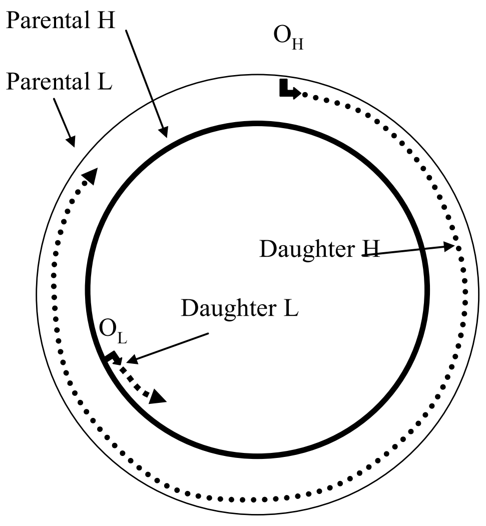


Figure 8-1. The parental H-strand is left single-stranded for an extended period of time during mtDNA duplication

Mitochondrial DNA is prone to damage from free oxygen radicals that occur during the production of ATP through the electron transport chain. Spontaneous deamination (Figure 8-2) of both A and C (Lindahl, 1993; Sancar and Sancar, 1988) occurs frequently in human mitochondrial DNA (Tanaka and Ozawa, 1994). Deamination of A leads to hypoxanthine that forms stronger base pair with C than with T, generating an A/T→G/C mutation. Deamination of C leads to U, generating C/G→U/A mutations.

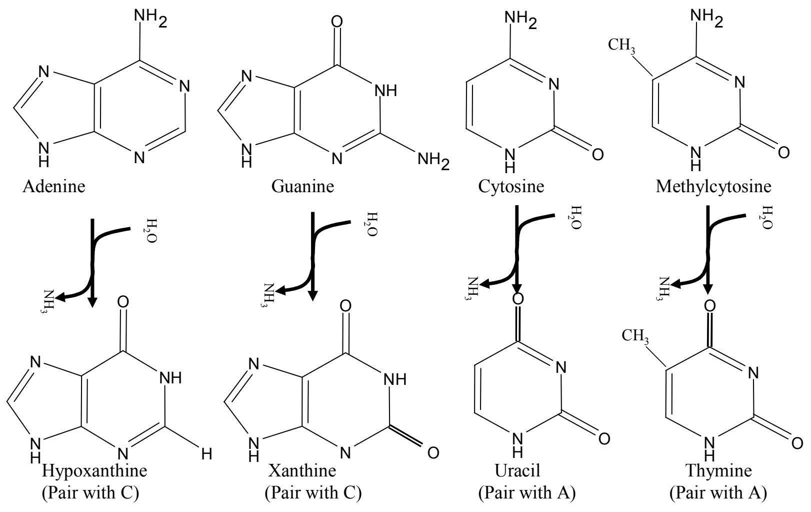


Figure 8-2. Spontaneous deamination of nucleotides.

Among these two types of spontaneous deamination, the C→U mutation occurs more frequently than the A→G mutation (Lindahl, 1993). In particular, the C→U mutation mediated by the spontaneous deamination occurs in single-stranded DNA more than 100 times as frequently as double-stranded DNA (Frederico *et al.*, 1990). This implies that nucleotide C on the H strand, which is left single-straded for hours, is prone to mutation to U. Note that these C→U mutants will immediately be used as a template to replicate the daughter L-strand, leading to a G→A mutation in the L-strand after one round of DNA duplication. Therefore, the H-strand, left single-stranded for an extended period during DNA replication, tend to accumulate A→G and C→U mutations and become rich in G and T while the L-strand will become rich in A and C.

Table 8-1 lists nucleotide frequency distributions among human mitochondrial genomes. Two contrasts can be made, the first between the 14 tRNA genes collinear with the L-strand and the eight tRNA genes collinear with the H-strand, and the second between the 12 protein-coding genes collinear with the L-strand and the ND6 gene collinear with the H-strand. Those genes, be it tRNA-coding or protein-coding, collinear with the L-strand are significantly more AC-rich than those collinear with the H-strand (Table 8-1). The two rRNA genes are collinear with the L-strand and their AC-richness is similar to tRNA genes collinear with the L-strand. This pattern is similar for all vertebrate mitochondrial genomes.

Table 8-1. Nucleotide frequency distribution among human mitochondrial genes. tRNAL and tRNAH are concatenated tRNA genes collinear with L-strand and H-strand, respectively. ND6 are collinear with the H-strand.

SeqName	L	P_A	P_C	P_G	P_T
(RNA genes)					
ssu-rRNA	954	0.3281	0.2652	0.1918	0.2149
lsu-rRNA	1558	0.3504	0.2561	0.1739	0.2195
tRNAL(1)	1025	0.3571	0.1990	0.1522	0.2917
tRNAH(2)	*552*	*0.2409*	*0.1431*	*0.2862*	*0.3297*
(Protein-coding genes)					
ND1	957	0.2853	0.3595	0.117	0.2382
ND2	1044	0.3123	0.3343	0.0967	0.2567
COX1	1542	0.2717	0.2996	0.1621	0.2665
COX2	684	0.2865	0.3129	0.1491	0.2515
ATP8	207	0.3865	0.3333	0.0628	0.2174
ATP6	681	0.2996	0.3377	0.1072	0.2555
COX3	781	0.2676	0.3188	0.1498	0.2638
ND3	346	0.2919	0.2948	0.1098	0.3035
ND4L	297	0.2828	0.3098	0.1212	0.2862
ND4	1378	0.3019	0.3454	0.0994	0.2533
ND5	1812	0.3035	0.3427	0.1065	0.2472
CYTB	1135	0.2881	0.3419	0.1172	0.2529
ND6	*525*	*0.1943*	*0.0724*	*0.3581*	*0.3752*

(1) pooled tRNA genes collinear with the L-strand (n = 14).
(2) pooled tRNA genes collinear with the H-strand (n = 8).

1.3 The effect of strand-biased mutation on codon usage

When transitional mutations (i.e., C↔U and G↔A) happen at the first or the second codon positions, they are mostly nonsynonymous. Such nonsynonymous mutations are typically purged off by purifying selection. However, if such mutations happen at the third codon position, then they are synonymous and tend to escape purifying selection and accumulate under the biased mutation pressure. Given the AC-biased mutation on the L-strand and GT-biased mutation on the H-strand, we may make the following predictions:

(1) Most codons in the 12 CDS sequences (that are collinear with the L-strand) end with A or C. Specifically, we should expect NNY codon families to be dominated by C-ending codons, NNR codon families dominated by A-ending codons, and NNN codons dominated by A-ending and C-ending codons.

(2) The codon bias in the ND6 gene on the opposite strand should be the opposite, and

(3) The 8 tRNA sequences collinear with the H-strand should be richer in G and T than the 14 tRNA sequences collinear with the L-strand.

These expectations have been confirmed by research on both eubacterial genomes (Lobry, 1996; Lobry and Sueoka, 2002; McInerney, 1998) and vertebrate mitochondrial genomes (Xia, 2005c). While many vertebrate mitochondrial DNAs (mtDNAs) are available, I will present data from only two teleost fish, *Erpetoichthys calabaricus* (GenBank accession: NC_005251) and *Masturus lanceolatus* (NC_005837) and two mammalian species, *Mus musculus* (NC_005089) and *Bos taurus* (NC_001567). There is no particular reason for choosing these species except for an effort to capture the rather limited diversity of vertebrate mitochondrial genomes.

The codon usage of the 12 CDS sequences collinear with the L-strand is consistent from this mutation bias, with the third codon position of the most frequent codon in each synonymous codon family (refereed to simply as codon family hereafter) being either A or C. In particular, NNY codon families are dominated by the C-ending codons, and NNR and NNN codon families are dominated by the A-ending codons (Table 8-2). While I present only data from the cow mtDNA, other vertebrate mtDNAs exhibit similar patterns. The remarkable consistency in this pattern from teleost fish to mammals demonstrates the power of the AC-biased mutation on the L-strand.

The observation that NNN codon families are dominated by NNA codons, instead by both NNA and NNC codons, might have adaptive significance (Xia, 1996) based on the observation that cellular concentration of ATP is much higher than that of the other three rNTPs (Colby and Edlin, 1970). For example, in the exponentially proliferating chick embryo fibroblasts in culture, the concentration of ATP, CTP, GTP and UTP, in the unit of (moles $\times 10^{-12}$ per 10^6 cells), is 1890, 53, 190, and 130, respectively, in 2-hour culture, and 2390, 73, 220, and 180, respectively, in 12-hour culture. The transcription hypothesis of codon usage (Xia, 1996) states that, with the high availability of A and relatively low availability of the other three rNTPs, the transcription efficiency can be increased by maximizing the use of A in the third codon position of protein-coding genes. However, I now believe that an alternative hypothesis, invoking differential mutations mediated by different nucleotide pools (Bebenek *et al.*, 1992 and references cited therein), may be a better explanation for the slightly more frequent A-ending codons than C-ending codons. Not only do the four ribonucleotides differ greatly in concentration, but the deoxyribonucleotides also differ greatly in concentration, with dATP being the most abundant in mitochondria. The abundance of dATP is expected to increase A-biased mutation rate during DNA replication. This would explain why NNA codons tend to be more frequent than NNC codons.

Table 8-2. Codon usage in the cow mtDNA. Two-fold C- or U-ending codons (top) are dominated by C-ending codons. Two-fold A- or G-ending codons (middle) are dominated by A-ending codons. Four-fold degenerate codons (bottom) are dominated by A-ending codons, followed by C-ending codons. Other vertebrate mitochondrial genomes exhibit similar patterns.

Codon	AA	Freq	RSCU	Codon	AA	Freq	RSCU
UGC	C	16	1.524	AUC	I	160	1.029
UGU	C	5	0.476	AUU	I	151	0.971
GAC	D	46	1.438	AAC	N	102	1.291
GAU	D	18	0.563	AAU	N	56	0.709
UUC	F	130	1.156	AGC	S	42	0.955
UUU	F	95	0.844	AGU	S	9	0.205
CAC	H	63	1.355	UAC	Y	72	1.125
CAU	H	30	0.645	UAU	Y	56	0.875
AGA	*	1	0.444	UUA	L	100	1.038
AGG	*	0	0	UUG	L	10	0.104
UAA	*	7	3.111	AUA	M	214	1.705
UAG	*	1	0.444	AUG	M	37	0.295
GAA	E	73	1.698	CAA	Q	79	1.837
GAG	E	13	0.302	CAG	Q	7	0.163
AAA	K	88	1.814	UGA	W	91	1.82
AAG	K	9	0.186	UGG	W	9	0.18
GCA	A	102	1.693	CGA	R	42	2.71
GCC	A	90	1.494	CGC	R	11	0.71
GCG	A	1	0.017	CGG	R	3	0.194
GCU	A	48	0.797	CGU	R	6	0.387
GGA	G	93	1.927	UCA	S	98	2.227
GGC	G	60	1.244	UCC	S	64	1.455
GGG	G	19	0.394	UCG	S	4	0.091
GGU	G	21	0.435	UCU	S	47	1.068
CUA	L	283	2.938	ACA	T	150	2
CUC	L	95	0.986	ACC	T	95	1.267
CUG	L	29	0.301	ACG	T	14	0.187
CUU	L	61	0.633	ACU	T	41	0.547
CCA	P	85	1.789	GUA	V	82	1.964
CCC	P	63	1.326	GUC	V	46	1.102
CCG	P	3	0.063	GUG	V	9	0.216
CCU	P	39	0.821	GUU	V	30	0.719

In contrast to the L-strand with mutations favoring A and C, the H-strand is expected to accumulate mutations in the opposite direction, i.e., favoring G and T. Consequently, we should predict that the third codon position of the ND6 gene, which is the only one of the 13 protein-coding sequences collinear with the H-strand in vertebrate mtDNA, should be dominated by either G or U. This prediction is also borne out by the empirical evidence (Table 8-3). In particular, NNY codon families are dominated by the U-ending codons, and NNR and NNN codon families are dominated by the G-ending codons.

Table 8-3. Codon frequencies for the two teleost species combined (Teleost) and the two mammalian species combined (Mammal) for the ND6 gene collinear with the H-strand in the mitochondrial genome. The most frequently used codons are bolded for large N.

Codon	AA	Teleost	Mammal	Codon	AA	Teleost	Mammal
AGA	*	1	0	AUA	M	12	8
AGG	*	0	0	AUG	M	10	14
UAA	*	0	2	AAC	N	1	1
UAG	*	1	0	**AAU**	**N**	**3**	**8**
GCA	A	9	3	CCA	P	2	1
GCC	A	2	1	CCC	P	2	0
GCG	A	5	3	CCG	P	2	0
GCU	**A**	**16**	**8**	**CCU**	**P**	**3**	**5**
UGC	C	0	0	CAA	Q	0	0
UGU	**C**	**4**	**5**	CAG	Q	1	1
GAC	D	1	0	CGA	R	1	1
GAU	**D**	**3**	**10**	CGC	R	0	0
GAA	E	2	8	CGG	R	6	0
GAG	**E**	**10**	**9**	CGU	R	3	1
UUC	F	4	3	AGC	S	1	1
UUU	**F**	**19**	**23**	**AGU**	**S**	**4**	**6**
GGA	G	9	12	UCA	S	2	3
GGC	G	3	2	UCC	S	0	1
GGG	**G**	**18**	**21**	UCG	S	2	2
GGU	G	14	17	**UCU**	**S**	**16**	**6**
CAC	H	1	0	ACA	T	0	3
CAU	H	0	0	ACC	T	0	1
AUC	I	0	1	ACG	T	3	2
AUU	**I**	**12**	**27**	**ACU**	**T**	**4**	**8**
AAA	K	0	2	GUA	V	6	10
AAG	K	0	3	GUC	V	2	2
CUA	L	4	2	GUG	V	17	9
CUC	L	0	0	**GUU**	**V**	**24**	**22**
CUG	L	4	0	UGA	W	4	4
CUU	**L**	**8**	**4**	UGG	W	4	5
UUA	**L**	**26**	**25**	UAC	Y	4	2
UUG	L	14	14	**UAU**	**Y**	**13**	**17**

The third codon position pairs with the first position (the wobble site) of the tRNA anticodon. Given what we know of the strand-biased mutation and the biased codon usage, can we predict what nucleotide should occupy the tRNA wobble site? There are three alternative hypotheses proposed (Xia, 2005c), one termed mutation hypothesis, the second codon-anticodon adaptation hypothesis and the third the wobble-versatility hypothesis.

2. THREE HYPOTHESES ON TRNA ANTICODON

It might help to gain a bit familiarity with the generic structure of the tRNA anticodon loop in order to better appreciate the three alternative hypotheses that will be detailed below. The anticodon in almost all tRNA sequences from all species share the regular feature of being flanked by two nucleotides on either side to form a loop that is held together by a stem (Figure 8-3). For example, the anticodon loop (AC loop) of tRNA^{Ala} in *M. musculus* is 24AUUGA**UU**UGC**AU**UCAAU40 where the starting and ending numbers indicate the position of the AC loop in the tRNA sequence (numbered from 1), with the anticodon (5'-UGC-3') flanked by two nucleotides on either side (bolded) to form a loop that is held together by a stem made of the first and the last four nucleotides. Such a regular AC loop and its anticodon can be easily identified by dynamic programming.

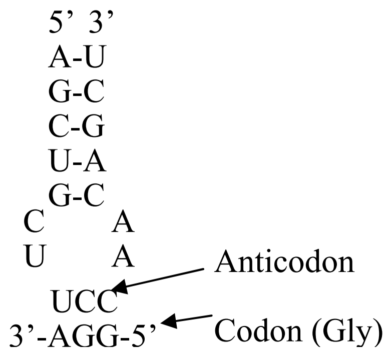


Figure 8-3. The stem-loop structure of the anticodon loop of tRNA^{Gly}. The anticodon 3'-AGG-5' is often written as GGA.

A few tRNA sequences have an anticodon flanked by three nucleotides, e.g., tRNA^{Val} in *Erpetoichthys calabaricus* and tRNA^{Ser1} in the blue whale, *Balaenoptera musculus*. Some tRNA sequences have a suspicious AC loop. For example, the AC loop of tRNA^{Trp} is 26GAGC**CU**UCA**AA**GCCC42 with a stem that has a mismatch. For such tRNA sequences with an irregular AC loop, one may identify AC loop by aligning the tRNA sequences against other isoaccepting tRNA sequences with a regular AC loop.

The two groups of Leu codons (CUN and UUR) are generally treated as two separate synonymous codon families because they have different tRNAs. The same applies to the two groups of Ser codons (AGY and UCN).

2.1 The mutation hypothesis

The strand-specific mutation bias is visible in RNA sequences (Table 8-1), with the 8 tRNA sequences (pooled) collinear with the H-strand being particularly rich in T and the two rRNA genes and the 14 tRNA sequences (pooled) collinear with the L-strand being particularly rich in A. This pattern is consistent from the teleost fish to mammalian species.

Given the strand-specific mutation bias in vertebrate mitochondrial genomes, what can we predict about the anticodon evolution of the tRNA sequences? In vertebrate mitochondrial genome, each tRNA anticodon essentially has to wobble in order to translate two or four synonymous codons. This suggests that the wobble position may not be strongly constrained and may be shaped by the strand-specific mutation bias. If the strand-specific mutation pressure is the dominant force in shaping anticodon evolution, then the 14 tRNA sequences collinear with the L-strand may have their wobble positions occupied by A, and the 8 tRNA sequences collinear with the H-strand may have their wobble positions occupied by U. This is the mutation hypothesis in a nutshell.

2.2 The codon-anticodon adaptation hypothesis

In contrast to the mutation hypothesis, the codon-anticodon adaptation is a selectionist hypothesis. Since the discovery of the correlation between codon usage and tRNA abundance in *Escherichia coli* (Gouy and Gautier, 1982; Ikemura, 1981) and *Saccharomyces cerevisiae* (Bennetzen and Hall, 1982), much progress has been made in understanding codon usage and codon-anticodon adaptation (Bulmer, 1987, 1991) in the context of maximizing transcription and translation rates (Akashi, 2003; Eyre-Walker, 1996; Xia, 1996, 1998b). In short, suppose two synonymous codons i and j, with codon i having many cognate tRNA to translate it and codon j having few or no cognate tRNA to translate it. In order to maximize translation efficiency in highly expressed protein-coding genes, we predict that natural selection should favor the use of codon i against codon j in highly expressed genes, leading to a strong association between frequently used codons and the abundance of their cognate tRNA. This prediction has been empirically substantiated numerous number of times.

Almost all publications consider mutation as disruptive to the evolution and maintenance of codon usage bias and the associated codon-anticodon adaptation (Akashi, 1995, 1997; Berg, 1996; Berg and Martelius, 1995; Xia, 1996). In other words, while selection is supposed to be the main force driving and maintaining the evolution of synonymous codons towards maximizing the codon that matches the anticodon of the most abundant tRNA, mutation is thought to reduce codon usage bias and disrupt codon-anticodon adaptation and is invoked whenever one fails to see strong codon usage bias or codon-anticodon adaptation. The relative abundance of different tRNA species is often, albeit implicitly, taken as prefixed, and this tRNA bias then drives codon usage bias. In spite of studies on the effect of mutation spectrum on GC content and amino acid usage (Lobry, 2004; Sueoka, 1961), there has been no empirical documentation of mutation pressure that maintains codon usage bias, and neither is there any report demonstrating that codon usage bias drives tRNA bias.

Given the codon usage bias in 12 of the 13 CDS sequences maintained by the strand-specific mutation pressure, it is easy to see from Table 8-2 that the overall codon usage bias at the genomic level is (1) C-ending codons most frequent in NNY codon families and (2) A-ending codons most frequent in NNR and NNN codon families. Such a codon usage may drive the wobble sites of the anticodon towards G for tRNA translating NNY codons or U for tRNA translating NNR or NNN codons, regardless of which strand the tRNA gene is on. Such anticodon evolution would increase the translation efficiency for the 12 protein-coding genes collinear with the L-strand but reduce the translation efficiency for the lone ND6 gene collinear with the H-strand. If the selection for translation efficiency is strong, then the 12 protein-coding genes would "out-vote" the lone ND6 gene.

2.3 The wobble versatility hypothesis

The wobble versatility hypothesis has been implicitly proposed before (e.g., Agris, 2004; Tong and Wong, 2004). Given that each synonymous codon family is translated by a single tRNA species in vertebrate mitochondria, the versatility of this single tRNA in translating two or four synonymous codons are important for the translation machinery. According to the conventional base-pairing rule first proposed for fungal mitochondria (Heckman *et al.*, 1980; Martin *et al.*, 1990), two-fold degenerate codons ending with C or U are translated by tRNA with a wobble G at its anticodon wobble site because G can pair not only with C, but also with U (Figure 8-4), two-fold degenerate codons ending with A or G are translated by tRNA with a wobble U at its anticodon wobble site because U can pair with both A and G, and four-fold degenerate codons are translated by tRNA with a wobble U at its anticodon wobble site. The versatility of U at the anticodon wobble site in pairing with other nucleotides has subsequently been substantiated in a number of species (Andachi *et al.*, 1989; Barrell *et al.*, 1980; Inagaki *et al.*, 1995; Sibler *et al.*, 1986; Yokobori *et al.*, 2001; Yokoyama and Nishimura, 1995).

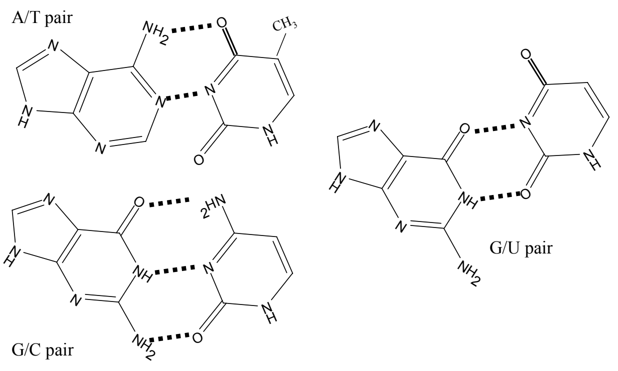


Figure 8-4. Canonical base-paring between nucleotides. The energy taken to break a G/U pair is roughly half of an A/T pair.

According to the wobble versatility hypothesis, the use of U and G at the wobble site of the tRNA anticodon can be predicted with no reference to codon usage, although codon-anticodon adaptation can then evolve as secondary adaptation given that wobble U and G are strictly maintained by selection for maximizing wobble versatility.

In summary, the mutation hypothesis predicts that the wobble site should most likely be A for tRNA genes collinear with the L-strand and T for tRNA collinear with the H-strand. The codon-anticodon adaptation hypothesis predicts that the wobble site should be G for NNY codon families to match the most abundant C-ending codon, and U for NNR and NNN codon families to match the most abundant A-ending codon. The wobble versatility hypothesis happens to have exactly the same prediction as the codon-anticodon adaptation hypothesis. It predicts that the wobble site should be G for NNY codon families because G can pair with both C and U, U for NNR codon families because U can pair with both A and G, and U for NNN families because U is the most versatile in wobbling. When two different hypotheses generate the same predictions, it becomes difficult to evaluate their respective validity. We will first evaluate the mutation hypothesis against the two selectionist hypotheses, and then make an attempt to evaluate the two selectionist hypotheses.

3. EMPIRICAL EVALUATION OF THE THREE ALTERNATIVE HYPOTHESES

3.1 Evaluation of the mutation hypothesis against the two selectionist hypotheses

Because the three hypotheses have explicit predictions of the wobble nucleotide in tRNA anticodons, a direct evaluation of the hypotheses is to compile the nucleotide at the tRNA anticodon wobble site to see which hypotheses provide the correct prediction. Empirical data (Table 8-4) strongly support the two selectionist hypotheses.

Table 8-4. Anticodon (AC) of the 22 tRNA genes from the four species and their associated synonymous codon families (SCF). "C" stands for "complementary strand", i.e., not on the same strand as the 12 protein-coding genes. Note that the first nucleotide of the anticodon (AC) is the wobble site.

tRNA	Strand	SCF	AC
Ala	C	GCN	UGC
Arg		CGN	UCG
Gly		GGN	UCC
Leu		CUN	UAG
Pro	C	CCN	UGG
Ser	C	UCN	UGA
Thr		CAN	UGU(1)
Val		GUN	UAC
Ser		AGY	GCU
His		CAY	GUG
Ile		AUY	GAU
Asn	C	AAY	GUU
Asp		GAY	GUC
Cys	C	UGY	GCA
Phe		UUY	GAA
Tyr	C	UAY	GUA
Gln	C	CAR	UUG
Glu	C	GAR	UUC
Leu		UUR	UAA
Lys		AAR	UUU
Met		AUR	CAU
Trp		UGR	UCA

(1) GGU in Mus musculus, which might be due to sequencing error because the anticodon loop is irregular.

There are two points worth highlighting in Table 8-4. First, for each tRNA, the anticodon is the same in all vertebrates from teleost fish to mammals. This implies that the selection at the wobble site must be very strong. Second, the wobble site is always G for tRNAs recognizing NNY codon families, and always U for tRNAs recognizing NNR and NNN codon families (with only one exception, involving tRNA^{Met}, which leads to another very interesting story told later). This is consistent regardless of which strand the tRNA sequence is on.

While the prediction of the selectionist hypotheses are consistent with the result, the prediction of the mutation hypothesis, that the wobble site should depend on which strand the tRNA gene is located, is clearly not supported (Table 8-4). Thus the mutation hypothesis can be readily rejected.

3.2 Evaluating the two selectionist hypotheses

The vertebrate mitochondrial data are unable to distinguish between the wobble versatility hypothesis and the codon-anticodon adaptation hypothesis that I have mentioned before because both hypotheses have exactly the same predictions for the anticodon wobble site. This illustrates a case in which the same observation can have two different interpretations both being entirely consistent with the observation. We see this often in real life. A man running after another may be interpreted as one in full pursuit of the other or as both trying to catch a departing train. We need more information to discriminate between these two interpretations. What information do we need to discriminate between the codon-anticodon adaptation hypothesis and the wobble versatility hypothesis?

The codon usage of the four-fold degenerate arginine codons (CGN) in the mitochondrial genome of four species: *Caenorhabditis elegans* (nematode), *Marchantia polymorpha* (plant), *Pichia canadensis* (fungi), and *Saccharomyces cerevisiae* (fungi) sheds light on resolving these two hypotheses (Table 8-5). In these four mitochondrial genomes, the four synonymous CGN codons, with CGU being the most dominant (Table 8-5), are translated by a single tRNA just as in vertebrate mitochondria. The wobble versatility hypothesis would have predicted a "versatile" U in the tRNA anticodon wobble site for these four-fold degenerate codons. However, this is not true because the wobble anticodon site is A instead of U in all four mitochondrial genomes. On the other hand, given that the CGU codon is the most dominant of the four synonymous arginine codons in all four mitochondrial genomes, the hypothesis of anticodon adaptation would predict an A at the anticodon wobble site, which is true for all four species.

Table 8-5. Codon usage of the four-fold degenerate arginine codons in four species.

Species	Accession	CGA	CGC	CGG	CGU
C. elegans	NC_001328	1	0	1	29
M. polymorpha	NC_001660	260	165	118	286
P. canadensis	NC_001762	0	1	0	19
S. cerevisiae	NC_001224	0	2	1	18

It is important to highlight the fact that none of the species in Table 8-5 is a vertebrate. Even if my interpretation of the result in Table 8-5 is correct, it is not necessarily generalizable to vertebrates. Furthermore, the limited result does not conclusively reject the wobble versatility hypothesis.

Scientists have a tendency to hold on to their own petty hypothesis and to discredit the hypothesis of others. However, it often happens that, when people of the two different camps learn to appreciate each other, the two seemingly incompatible hypotheses suddenly begin to merge into a single hypothesis. The codon-anticodon adaptation hypothesis and the wobble versatility hypothesis serve as a good example of two different hypotheses integrating into each other to form a more general hypothesis outlined in the next section.

4. INTEGRATING THE CODON-ANTICODON ADAPTATION HYPOTHESIS (CAAH) AND THE WOBBLE VERSATILITY HYPOTHESIS (WVH)

Here we develop a general hypothesis of codon-anticodon adaptation. Let's be more explicit on the cost of wobble translation. Define

$C_{U\text{-}G}$: cost of wobble pairing between U and G,
C_O: cost of wobble pairing other than the two above,

Let's also define the cost of perfect Watson-Crick paring as 0. Given that U is more versatile at wobbling than other nucleotides, we expect $C_O > C_{U\text{-}G} > 0$. The general hypothesis assumes that natural selection will drive the codon-anticodon coevolution in such a way as to minimize the total wobble cost (designated C_w) in translating a protein molecule. Below we derive predictions from the general hypothesis (refereed to as GH hereafter) and compare them with those from the codon-anticodon adaptation hypothesis (referred to as CAAH hereafter) and the wobble versatility hypothesis (referred to as WVH hereafter).

4.1 Four-fold NNN codons

Consider first a single four-fold codon family. Designate the number of codons ending with A, C, G, and U as N_A, N_C, N_G, and N_U, respectively. The wobble costs involving an A, C, G or U at the wobble site are, respectively,

$$\begin{aligned}
C_{wA} &= N_A C_O + N_C C_O + N_G C_O + N_U \times 0 \\
&= C_O(N_A + N_C + N_G) \\
C_{wC} &= C_O(N_A + N_C + N_U) \\
C_{wG} &= C_O(N_A + N_G) + N_U C_{U-G} \\
C_{wU} &= C_O(N_C + N_U) + N_G C_{U-G}
\end{aligned} \tag{8.1}$$

Eq. (8.1) provides a means of estimating the relative magnitude of C_O, $C_{U\text{-}G}$. For example, we may replace N_A, N_C, N_G and N_U by *M. polymorpha* data in Table 8-5 to get

$$\begin{aligned}
C_{wA} &= C_O(260 + 165 + 118) \\
C_{wC} &= C_O(260 + 165 + 286) \\
C_{wG} &= C_O(260 + 118) + 286 C_{U-G} \\
C_{wU} &= C_O(165 + 286) + 118 C_{U-G}
\end{aligned} \tag{8.2}$$

We can see immediately that $C_{wC} > C_{wA}$ because 286 > 118. Because we know that nature has chosen nucleotide A at the anticodon wobble site of the tRNA, C_{wA} should also be smaller than C_{wG} and C_{wU}. This gives us two inequalities. From $C_{wA} < C_{wG}$, we have

$$\begin{aligned}
&C_O(260 + 165 + 118) < C_O(260 + 118) + 286 C_{U-G} \\
&165 C_O < 286 C_{U-G} \\
&\frac{C_O}{C_{U-G}} < \frac{286}{165}
\end{aligned} \tag{8.3}$$

From $C_{wA} < C_{wU}$, we have

$$\begin{aligned}
&C_O(260 + 165 + 118) < C_O(165 + 286) + 118 C_{U-G} \\
&\frac{C_O}{C_{U-G}} < \frac{118}{92}
\end{aligned} \tag{8.4}$$

Eq. (8.4) suggests that the cost associated with wobble pairing may be quite small, roughly the same as pairing between U and G. Given this, it becomes easy to understand that the tRNA for translating the CGN codon family (Table 8-5) should have nucleotide A at its wobble site in *C. elegans, P. Canadensis* and *S. cerevisiae* because C_{wA} is by far the smallest.

4.2 Two-fold NNY codon families

Designate C-ending and U-ending codons by N_C and N_U, respectively, and the total cost of wobble pairing as C_{wG} when the wobble site of the anticodon is G and as C_{wA} when the wobble site is A (we do not need to consider the case when the wobble site is U or C because such cases have never been observed and because neither wobble versatility hypothesis nor codon-anticodon adaptation hypothesis would predict a wobble site that is C or U in NNY codon families). We can now express C_{wG} and C_{wA} as

$$\begin{aligned} C_{wG} &= N_C \times 0 + N_U C_{U-G} \\ C_{wA} &= N_C C_O + N_U \times 0 \end{aligned} \tag{8.5}$$

The multiplication by 0 in Eq. (8.5) arises from our definition that perfect Watson-Crick pairing has zero wobble cost. Now we consider three special cases.

First, if $N_C > N_U$, then $C_{wG} < C_{wA}$, and GH predicts that the anticodon wobble site should be occupied by a G. This prediction is shared by both CAAH and WVH.

Second, if $N_C = N_U = N/2$, then Eq. (8.5) is reduced to

$$\begin{aligned} C_{wG} &= N_U C_{U-G} = \frac{N}{2} C_{U-G} \\ C_{wA} &= N_C C_O = \frac{N}{2} C_O \end{aligned} \tag{8.6}$$

Because $C_{U\text{-}G} < C_O$, we have $C_{wG} < C_{wA}$, and GH predicts that the anticodon wobble site should be a G. This is the same prediction as WVH. In this case, CAAH has no prediction.

Third, when $N_C << N_T$, especially in the extreme case when $N_C = 0$ and $N_T = N$, then Eq. (8.5) is reduced to

$$\begin{aligned} C_{wG} &= NC_{U-G} \\ C_{wA} &= 0 \end{aligned} \tag{8.7}$$

Because $C_{wG} > C_{wA}$, GH predicts an A at the anticodon wobble site. In this case, WVH would still predict a G at the wobble site because it ignores the codon frequencies, but CAAH would predict an A at the wobble site, which is the same prediction as GH. Only in this particular case when $N_C << N_T$ can CAAH and WVH be clearly differentiated.

4.3 Two-fold NNR codon families

Following the same reasoning above, we can come to the following conclusions. When $N_A > N_G$, then GH, CAAH and WVH all have the same prediction that the anticodon wobble site should be a U. When $N_A = N_G$, then $C_{wU} < C_{wC}$, and GH predicts a U at the anticodon wobble site, the same prediction as WVH.. In this case, CAAH has no specific prediction. In the extreme case when $N_A = 0$ and $N_G = N$, then

$$\begin{aligned} C_{wU} &= NC_{U-G} \\ C_{wC} &= 0 \end{aligned} \tag{8.8}$$

Because $C_{wC} < C_{wU}$, GH predicts a C at the anticodon wobble site. This is the same prediction as that of CAAH but different from that of WVH which predicts a U at the anticodon wobble site. Only in this particular case can CAAH and WVH be clearly differentiated.

5. CONFLICT BETWEEN TRANSLATION INITIATION AND ELONGATION

By now we have resolved almost all controversies except for one little puzzle. We have previously noted an exception in Table 8-4. All tRNA translating two-fold codon families ending with A or G has a nucleotide U at its anticodon wobble site except for $tRNA^{Met}$. The $tRNA^{Met}$ anticodon is 3'-UAC-5' (or CAU for short), with the wobble site being C instead of U, and forms a Watson-Crick match with the AUG codon instead of the AUA codon, in spite of the fact that the latter is used much more frequent than the former. The ability of the CAU anticodon to pair with the AUA codon is achieved by modifying the C in the anticodon CAU to 5-formylcytidine (Matsuyama *et al.*, 1998; Moriya *et al.*, 1994). A similar case involves the methylation of guanine in starfish tRNA^{Ser} to translate all four AGN codons (Matsuyama *et al.*, 1998).

The use of the CAU anticodon instead of a UAU anticodon in vertebrate mitochondrial tRNA^{Met} is unexpected from two existing hypotheses of anticodon usage, i.e., CAAH and WVH, mentioned before. CAAH predicts that the anticodon should match the most abundant codon, i.e., AUA instead of AUG. So the anticodon should be UAU instead of the observed CAU. WVH (Agris, 2004; Tong and Wong, 2004; Xia, 2005c) states that the anticodon should maximize its wobble versatility in paring with synonymous codons. Because U can pair with both A and G, this hypothesis also predicts an UAU anticodon to maximize its paring versatility with the AUA and AUG codons. The fact that the observed tRNA^{Met} anticodon is CAU instead of the predicted UAU is intriguing. After all, why not just use the UAU anticodon that can pair with both AUA and AUG codons instead of having a CAU anticodon and then chemically modifying it to pair with AUA codons?

This unexpected tRNA^{Met} anticodon has been attributed to a compromise between translation initiation and elongation (Xia, 2005c) as follows. AUG is not only the most frequently used initiation codon, but also the most efficient initiation codon in *Escherichia coli* (Romero and Garcia, 1991) and *Saccharomyces cerevisiae* (Nett *et al.*, 2001). In *E. coli,* the most efficient non-AUG initiation codon is AUA and its rate of initiation is only 7.5% of AUG (Romero and Garcia, 1991). In yeast mitochondria, a mutation of the initiation AUG to AUA in the COX2 gene caused at least a five-fold decrease in translation (Mulero and Fox, 1994), and similar finding was also duplicated in another yeast mitochondrial gene COX3 (Folley and Fox, 1991). Assuming the generality of these findings, an anticodon matching AUG will increase the initiation rate and would be favored by natural selection because translation initiation is often the limiting step in protein production (Bulmer, 1991; Liljenstrom and von Heijne, 1987). This presents a conflict between translation initiation and translation elongation. An AUG-matching anticodon would increase translation initiation rate but decrease translation elongation rate because an overwhelming majority of methionine codons are AUA in vertebrate mitochondrial genomes. The fact that all known vertebrate tRNA^{Met} genes feature an AUG-matching codon implies that nature has chosen to maximize the translation initiation rate (Xia, 2005c). This hypothesis that invokes a conflict between translation initiation and translation elongation to explain the usage of the CAU anticodon in tRNA^{Met} will be referred hereafter as the translation conflict hypothesis.

Two consequences can be derived from the translation conflict hypothesis. First, we should expect a relative reduction of AUA usage because the AUG-matching anticodon imposes selection against the use of AUA codons as AUA would need to be wobble-translated by a chemically modified CAU anticodon. To fix ideas, let us focus only on AUR (methionine) and UUR (leucine) codon families. The reason for choosing

UUR instead of any other XYR codon families is because other XYR codon families do not have a middle U and the middle nucleotide in a codon is known to affect the nucleotide at the third codon position.

For the 12 CDSs that are collinear with the AC-rich L-strand, the mutation favors A-ending codon (Reyes *et al.*, 1998; Tanaka and Ozawa, 1994; Xia, 2005c). For UUR codons, because the anticodon wobble site is U and form Watson-Crick base pair with A, we also expect UUA codon to be preferred against UUG codons. Thus, both mutation and the tRNA-mediated selection favor the use of UUA against UUG codons. However, for the methionine codons, the AUG-matching $tRNA^{Met}$ anticodon would favor the AUG codon against the AUA codon. Thus, the tRNA-mediated selection and the mutation bias are in opposite directions. If we define

$$P_{XUA} = \frac{100N_{XUA}}{N_{XUA} + N_{XUG}} \qquad (8.9)$$

for each of these two codon families, where N_{XUA} and N_{XUG} are the number of XUA and XUG codons, respectively, we should find P_{AUA} to be smaller in the AUR codon family than P_{UUA} in the UUR codon families.

An argument against using Eq. (8.9) is that the result would be biased in favor of supporting the prediction of $P_{AUA} < P_{UUA}$ because the initiation codon, which is AUG in most cases, was not excluded. A more convincing comparison should compute P_{AUA} after excluding initiation codons entirely. This is what we are going to use.

For the ND6 gene collinear with the GT-rich H-strand, the strand-biased mutation spectrum favors G-ending codons in the two XUR codon families. For the methionine codon family, the AUG-matching anticodon also favors the AUG codon against the AUA codon. So the AUA codon will be depressed by both the strand-biased mutation and the tRNA-imposed selection. The tRNA-imposed selection is absent against UUA codon in the UUR codon families because their respective tRNA anticodons all match the A-ending codons (Xia, 2005c). Thus, for the ND6 gene, we also expect P_{AUA} to be smaller in the AUR codon family than P_{UUA} in the UUR codon families.

Many vertebrate mitochondrial genomes can be used to test the prediction that P_{UUA} should be greater than P_{AUA}. For ease of presentation, we will use only 30 vertebrate species covering a wide range of taxonomic diversity. These include six each from mammals, birds, reptiles (now a nearly obsolete taxonomic term), amphibians and fish. Results from these species, summarized in Table 8-6, substantiates the predictions based on the translation conflict hypothesis.

Table 8-6. Results from the 12 CDS sequences collinear with the L-strand and ND6 collinear with the H-strand from 30 representative vertebrate species.

		12 CDS		ND6	
Species	Accession	P_{UUA}	P_{AUA}	P_{UUA}	P_{AUA}
Homo sapien	NC_001807	87.8	87.1	53.3	22.2
Mus musculus	NC_005089	94.2	93.4	65.2	40.0
Bos Taurus	NC_006853	90.1	87.8	62.5	40.0
Canis familiaris	NC_002008	85.3	85.5	43.8	50.0
Equus caballus	NC_001640	86.4	87.5	25.0	11.1
Capra hircus	NC_005044	94.4	88.5	68.8	66.7
Gallus gallus	NC_001323	94.9	88.2	31.6	25.0
Struthio camelus	NC_002785	89.2	87.9	27.3	0.0
Coturnix chinensis	NC_004575	95.7	88.0	28.6	0.0
Anser albifrons	NC_004539	92.2	82.9	41.7	0.0
Gavia stellata	NC_007007	88.0	91.9	7.1	33.3
Alectura lathami	NC_007227	83.1	80.4	61.1	25.0
Alligator mississippiensis	NC_001922	94.2	84.4	64.7	33.3
Alligator sinensis	NC_004448	78.2	76.1	45.5	37.5
Chelonia mydas	NC_000886	99.1	98.6	30.4	75.0
Shinisaurus crocodilurus	NC_005959	94.2	89.0	42.9	33.3
Abronia graminea	NC_005958	92.0	91.2	42.9	12.5
Chrysemy picta	NC_002073	92.7	94.5	50.0	33.3
Ambystoma laterale	NC_006330	95.0	91.5	91.3	71.4
Aneides hardii	NC_006338	92.6	85.4	45.0	77.8
Xenopus laevis	NC_001573	93.6	88.6	64.7	75.0
Kaloula pulchra	NC_006405	87.9	76.7	47.8	0.0
Alytes obstetricans	NC_006688	90.5	80.1	57.1	0.0
Rana nigromaculata	NC_002805	91.3	69.1	66.7	40.0
Cyprinus carpio	NC_001606	97.7	79.1	62.5	20.0
Danio rerio	NC_002333	89.6	78.9	57.1	50.0
Salanx ariakensis	NC_006918	73.9	46.9	77.8	0.0
Carassius auratus	NC_002079	93.9	75.5	60.0	40.0
Anguilla rostrata	NC_006547	89.1	83.2	57.9	66.7
Auxis rochei	NC_005313	82.8	38.8	83.3	12.5

Table 8-6 shows that mean P_{UUA} is significantly greater than mean P_{AUA} in both the 12 CDSs collinear with the L-strand (90.32 versus 82.56, DF = 29, T = 4.256, p = 0.0000, one-tailed test) and the ND6 gene (52.12 versus 33.05, DF = 29, T = 3.762324119, p = 0.0004, one-tailed test). In short, the prediction that AUA codon usage is reduced is empirically supported by genes from both DNA strands. We may conclude that the first prediction, that AUA codon usage should be reduced given the CAU anticodon in $tRNA^{Met}$, is generally supported by the empirical analysis.

The observation of a relative deficiency of AUA codons can be interpreted in two ways. If methionine usage remains constant among vertebrate mitochondrial genomes, then a deficiency of AUA in a genome implies an equal amount of surplus in AUG. On the other hand, if the number of methionine codons (N_{Met}) is weakly constrained, then the selection against AUA codons may result in a net loss of methionine codons. This would lead to a positive association between P_{AUA} and N_{Met}, i.e., small P_{AUA} is associated with small N_{Met}. The empirical data supports the latter inference, i.e., reduction of AUA codons leads to a reduction in methionine usage in the genome (Figure 8-5).

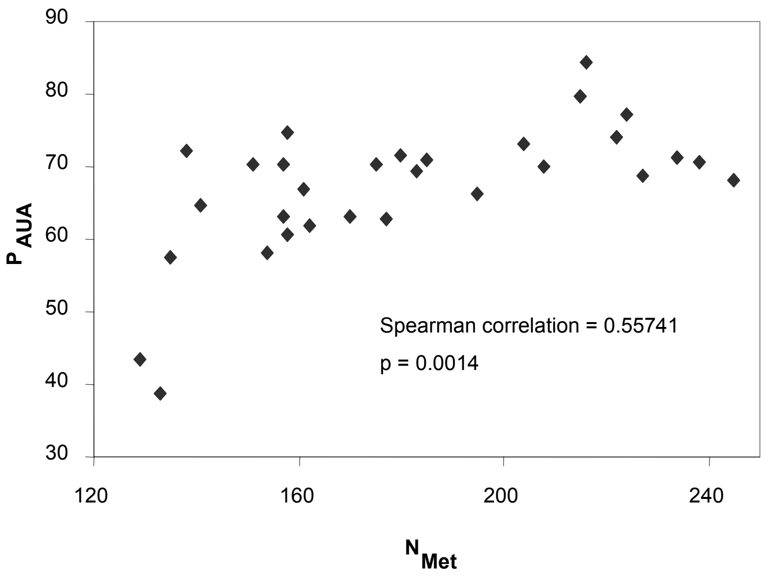


Figure 8-5. Genomic reduction of AUA codons is associated with a reduction in methionine usage. P_{AUA} is defined in equation and arcsine-transformed; N_{Met} – Number of methionine codons.

It is important to recognize that the results presented above, while all consistent with the translation conflict hypothesis, do not exclude the possibility that AUA codon usage may be reduced for reasons unrelated to the CAU anticodon in $tRNA^{Met}$. It would be nice to have a mitochondrial genome in which the $tRNA^{Met}$ anticodon is not CAU but UAU. If such a genome also has a reduced AUA usage relative to UUA codons, then we cannot interpret the reduced AUA usage in the vertebrate mitochondrial genomes as a response to the selection mediated by the CAU anticodon in $tRNA^{Met}$. On the other hand, if such a genome does not exhibit a deficiency of AUA codons relative to UUA codons, but instead exhibit an increased AUA codon usage favored by the UAU anticodon, then the translation conflict hypothesis is strengthened.

It is also important to keep in mind that the 30 species above do not represent independent data points. For example, their common ancestor could have somehow evolved a reduced P_{AUA} relative to P_{UUA}, and this character has been inherited among all its descendents. This means that all 30 species could be equivalent to just single data point. For this reason, corroborative evidence needs to be sought in other species.

In this context the mitochondrial genomes of four urochordates (*Halocynthia roretzi, Ciona intestinalis, C. savignyi,* and *Doliolum nationalis*) deposited in GenBank are particularly useful in providing corroborative evidence. All four genomes have two $tRNA^{Met}$ genes, one with CAU anticodon and the other with a UAU anticodon (Gissi *et al.*, 2004; Hoffmann *et al.*, 1992; Kondow *et al.*, 1998; Yokobori *et al.*, 2005; Yokobori *et al.*, 1999; Yokobori *et al.*, 2003). This would eliminate the hypothesized selection against AUA codon usage. We can therefore predict that P_{AUA} should not be underused relative to P_{UAU} for these urochordate genomes, in contrast to the vertebrate mitochondrial genomes in which the only $tRNA^{Met}$ has a CAU anticodon that would favor a decreased usage of AUA codons. In other words, we should expect $P_{UUA} - P_{AUA}$ to be near 0, in contrast to vertebrate mitochondrial genomes where $P_{UUA} - P_{AUA}$ is generally greater than 0. This prediction is confirmed (Table 2). The mean $P_{UUA} - P_{AUA}$ is only -3.225, in contrast to the vertebrate mitochondrial genome where the mean $P_{UUA} - P_{AUA}$ values are significantly greater than 0 ($p = 0.0001$ for the 12 CDSs collinear with the L-stand and $p = 0.0004$ for ND6 collinear with the H-strand).

Table 8-7. Results from the 13 CDSs from the four urochordate species.

Species	ACCESSION	PUUA	PAUA	P_{UUA}-P_{AUA}
H. roretzi	NC_002177	60.7	67.3	-6.6
C. intestinalis	NC_004447	92.6	90.5	2.1
C. savignyi	NC_004570	75.1	83.5	-8.4
D. nationalis	NC_006627	71.0	71.0	0.0

In conclusion, the translation conflict hypothesis is empirically supported. The presence of a CAU anticodon matching the AUG methionine codon represents a significant selection force against AUA codon usage in vertebrate mitochondrial genomes, resulting in P_{AUA} smaller than P_{UUA}. The reduced AUA codon usage is associated with a reduced methionine usage in the vertebrate mitochondrial genomes. When such selection is weakened in the urochordate mitochondrial genomes containing CAU-$tRNA^{Met}$ and UAU-$tRNA^{Met}$, the AUA codon is no longer strongly selected against, and P_{AUA} becomes similar to P_{XUA}.

I should finally mention here that, although the conceptual framework that leads to the prediction of strand-asymmetry in mutation spectrum is based on the classical strand-displacement model of mtDNA replication

(Bogenhagen and Clayton, 2003; Clayton, 1982; Shadel and Clayton, 1997) the prediction can also be derived from the strand-coupled model of bidirectional mtDNA replication (Holt and Jacobs, 2003; Holt *et al.*, 2000; Yang *et al.*, 2002). Many studies have documented an excess of (G+T) in the leading strand and an excess of (A+C) in the lagging strand in most prokaryotic genomes examined (Francino and Ochman, 1997; Freeman *et al.*, 1998; Grigoriev, 1998; McLean *et al.*, 1998; Perriere *et al.*, 1996) and spontaneous deamination has also been invoked as the main factor contributing to the strand asymmetry (Lobry and Sueoka, 2002). Thus, if the H-strand is the leading strand, and the L-stand the lagging one, then we would also predict an excess of (G+T) in the H-strand and of (A+C) in the L-strand, just as we would expect from the strand-displacement model of mtDNA replication.

One limitation of this study is that it cannot be generalized to invertebrate mitochondrial genomes although they also have about 13 protein-coding genes and 22 tRNA genes. There are several major differences between vertebrate and invertebrate mtDNA. First, invertebrate mitochondrial genomes are generally extremely AT-rich and the distribution of the protein-coding genes is less asymmetrical between the two strands than in vertebrate mitochondrial genomes. Take the common honey bee mtDNA for example. Nine of the 13 CDSs are collinear with the L-strand and 4 are collinear with the H-strand, in contrast to 12 CDSs collinear with the L-strand and only 1 collinear with the H-strand in vertebrate mitochondrial genomes.

6. POSTSCRIPT

In both Chapter 4 and this chapter, I have tried to illustrate the value of thinking from an evolutionary point of view. I did this because there are two contrasting and radical views of evolution in science that harms biology in general and bioinformatics in particular. In one extreme, many biologists, especially traditional molecular biologists, regard evolutionary biology as totally irrelevant and have little hesitation to proclaim “what is true for the colon bacillus is true for the elephant” (attributed to Jacques Monod, Jacob, 1988, p. 290). In the other extreme, many evolutionary biologists consider evolution as the key to every problem in biology and indulge themselves with the assertion that “Nothing in biology makes sense except in the light of evolution” (Dobzhansky, 1973). Some have gone even further by claiming that “in a sense all evolution is adaptation” (Medawar and Medawar, 1983, p. 1).

Extreme views, especially when taken out of the context in which they are formed, would often become so radical as to serve no purpose other than brainwashing the unwary and thwarting healthy communication that is so essential for science, especially for the normal development of the interdisciplinary bioinformatics.

Historians have often been criticized for not taking averages, but why neither do biologists?

Chapter 9

CHARACTERIZING TRANSLATION EFFICIENCY

Indices of codon usage

1. INTRODUCTION

The genome comes alive mainly through transcription and translation. The previous chapter on Gibbs sampler can be used to discover sequence motifs that serve as genetic switches regulating transcription and translation. In this chapter, we focus on a few simple but useful indices that can be used to measure the efficiency of translation elongation. You are expected to have read Chapter 8 and have already gained familiarity with biological concepts involving codon, tRNA, and anticodon in tRNAs.

Many unicellular organisms, especially bacterial species, need to grow and replicate the cell rapidly in order not to be out-competed by others. For example, an *E. coli* cell replicates once every 20 minutes with unlimited nutrients. To replicate a cell, not only the genome needs to be replicated, but a large amount of proteins have to be produced, with some proteins produced in nearly half a million copies in an *E. coli* cell. For such highly expressed proteins, it is very important for their coding genes to have efficient coding strategy to maximize the rate of transcription and translation.

One way to increase the translation efficiency is to maximize the usage of codons that match the anticodon of the most abundant cognate tRNA (Gouy and Gautier, 1982; Ikemura, 1992; Xia, 1998b, 2005c). For example, the amino acid glycine can be coded by GGA, GGC, GGG and GGU codons, but tRNA^{Gly} species that translates GGY codons (Y stands for either C or U) are much more abundant than tRNA^{Gly} species that translate GGR codons in

E. coli cells. What codons should *E. coli* use to code glycine? Obviously natural selection should favor those that maximize the usage of CCY codons against GGR codons given the differential tRNA availability.

There are two different notations of tRNA. For example, a glycine tRNA may be written either as GGY-tRNA^{Gly} where GGY is the codons that the tRNA can translate, or ACC-tRNA^{Gly} where ACC is the tRNA anticodon written in the 5' to 3' direction. The first notation is often used when a tRNA species is known to carry a glycine and to translate GGY codons, but the tRNA sequence and consequently the anticodon are unknown. The second notation is used when the anticodon is known. The two notations do not cause confusion because, given the tRNA^{Gly} part of the notation, it is unlikely for one to take ACC as a glycine codon.

Translation efficiency depends partially on the coding strategy of an mRNA and is reflected in codon usage bias which is often measured by two classes of indices, one class being codon-specific and the other being gene-specific. A representative of the first class is the relative synonymous codon usage or RSCU (Sharp *et al.*, 1986), and a representative of the second class is the codon adaptation index, or CAI (Sharp and Li, 1987).

Other than CAI, several other indices have been proposed to measure codon usage bias of protein-coding genes. All these indices (including CAI) measure codon usage bias in two ways. One is to measure the deviation of codon usage from random expectation or from equal codon usage. The random expectation can be derived as follows. Designating the genomic nucleotide frequencies as P_A, P_C, P_G and P_T, respectively, the expected probability of a codon, e.g., ACG, is simply $P_AP_CP_G$. A representative of this type of codon usage indices is the effective number of codons (Wright, 1990) which measures codon usage bias by the deviation of codon usage from equal codon usage.

The other codon usage indices measure codon usage bias by their degree of using translationally favored codons. They differ fundamentally in how they define translationally favored codons. The frequency of optimal codons, or F_{op} (Ikemura, 1985), defines translationally optimal codons as those forming Watson-Crick base pair with the anticodon of major tRNA species in each codon family. The codon adaptation index (CAI) defines translationally optimal codons as those frequently represented in highly expressed genes. The codon bias index, or CBI (Bennetzen and Hall, 1982) defines translationally favored codons as those not only frequently represented by highly expressed genes but also forming Watson-Crick base pair with the anticodon of major tRNA species. Comparative studies (Coghlan and Wolfe, 2000; Comeron and Aguade, 1998) suggest that CAI is the best in predicting gene expression levels. For this reason, we will only detail the computation and application of CAI. Readers interested in other indices may read the original publications.

It is natural for one to ask why we should not use indices derived directly from relative tRNA abundance or why such indices often do not perform better than CAI which ignores tRNA but is based entirely on the codon usage of highly expressed genes. We will provide answers to these questions once we know how to compute RSCU and CAI.

2. RSCU (RELATIVE SYNONYMOUS CODON USAGE)

RSCU measures codon usage bias for each codon within each codon family. It is essentially a normalized codon frequency so that the expectation is 1 when there is no codon usage bias. A codon is overused if its RSCU value is greater than 1 and underused if its RSCU value is less than 1. It is computed directly from input sequences.

The general equation for computing RSCU is

$$RSCU_{ij} = \frac{CodFreq_j}{\dfrac{\left(\sum_{j=1}^{NumCodon_i} CodFreq_i \right)}{NumCodon_i}} \tag{9.1}$$

where i refers to a codon family and j to a specific codon within the family. For example, i may refer to the alanine codon family with four codons (GCU, GCC, GCA, and GCG) and j to a specific codon such as GCU. In this case, the numerator is the frequency of GCU and denominator is the summation of the four codon frequencies divided by the number of codons in the codon family, i.e., 4.

For biology students, it is always easier to learn by numerical examples. Suppose we counted the codon frequencies of one particular protein-coding sequence and have obtained the codon frequencies (Table 9-1). The RSCU for the GCU codon is computed, according to Eq. (9.1), as

$$RSCU_{GCU} = \frac{52}{\dfrac{(52+91+103+2)}{4}} = 0.84 \tag{9-2}$$

which is displayed in Table 9-1. Biology students are recommended to cover up the last column in Table 9-1 and finish the computation of the rest of the RSCU values.

Table 9-1. Data for illustrating the calculation of RSCU. AA-amino acid; T-codon frequency.

Codon	AA	N	RSCU
GCU	Ala	52	0.84
GCC	Ala	91	1.47
GCA	Ala	103	1.66
GCG	Ala	2	0.03
GAA	Glu	78	1.64
GAG	Glu	17	0.36
...	...	...	...

3. CAI (CODON ADAPTATION INDEX)

CAI has been used extensively in biological research. Other than its primary use for measuring the efficiency of translation elongation, it has contributed to the finding that functionally related genes are conserved in their expression across different microbial species (Lithwick and Margalit, 2005), to the prediction of protein production (Futcher *et al.*, 1999; Gygi *et al.*, 1999), and to the optimization of DNA vaccines (Ruiz *et al.*, 2006).

3.1 Computation and basic properties of CAI

While RSCU characterizes codon usage bias in each codon family, CAI quantifies the codon usage bias in one gene. It is based on (1) the codon frequencies of the gene and (2) the codon frequencies of a set of known highly expressed genes (often referred to as the reference set). The reference set of genes is used to generate a column of w values computed as:

$$w_{ij} = \frac{RefCodFreq_{ij}}{RefCodFreq_{i.\max}} \tag{9-3}$$

where $RefCodFreq_{ij}$ is the frequency of codon j in synonymous codon family i, and $RefCodFreq_{i.max}$ is the maximum codon frequency in synonymous codon family i. For example, if the four alanine codons GCA, GCC, GCG and GCU have frequencies 20, 4, 4, and 2, respectively, then their associated w value are 1, 0.2, 0.2 and 0.1, respectively. The codon whose frequency is $RefCodFreq_{i.max}$ is often referred to as the major codon (whose w is 1), and the other codons in the synonymous codon family are referred to as minor codons. The major codon is assumed to be the translationally optimal codon.

It is easy to see the relationship between w_{ij} and RSCU. The former is obtained by dividing each RSCU by the largest RSCU value within each codon family. With the w values for a particular species, we can now compute the CAI value of any protein-coding sequence from the species by using the following equation:

$$CAI = e^{\left(\frac{\sum_{i=1}^{n}[CodFreq_i \ln(w_i)]}{\sum_{i=1}^{n} CodFreq_i} \right)} \qquad (9\text{-}4)$$

where n is the number of sense codons (excluding codon families with a single codon, e.g., AUG for methionine and UGG for tryptophan in the standard genetic code). Note that the exponent is simply a weighted average of ln(w). Because the maximum of w is 1, ln(w) will never be greater than 0. Consequently, the exponent will never be greater than 0. Thus, the maximum CAI value is 1. The minimum CAI depends on the w values for minor codons in each codon family. If the minor codons all have w values close to zero, then the minimum CAI will also be very close to zero.

A gene with a CAI value greater than 0.7 is often considered to be highly expressed (Sharp and Li, 1987; Takahashi *et al.*, 2003). However, this interpretation is questionable because the value of CAI depends on what reference set of genes is used. For the same reason, CAI values are not comparable between or among different species. One should never say something like “Gene A may be more highly expressed in human than in mouse because its CAI value is 0.8 in human and only 0.4 in mouse”. Such a statement is wrong in at least two ways. First, CAI values for mouse are computed with a set of mouse genes as reference and those for human are computed with a set of human genes as reference. They are not comparable. Second, CAI is not an index of gene expression, but an index of translation elongation efficiency. Strictly speaking, it is not even correct to say that “the mRNA from human gene A can be translated more efficiently than that from human gene B because the former has a CAI value twice as larger as the latter”. The reason is that the translation process involves initiation, elongation and termination. CAI is a measure of the efficiency of translation elongation, not that of translation initiation or termination.

One can also derive a similar amino acid adaptation index (AAAI) to measure the bias of amino acid usage:

$$AAAI = e^{\left(\frac{\sum_{i=1}^{n}[AAFreq_i \ln(w_i)]}{\sum_{i=1}^{n} AAFreq_i}\right)} \tag{9.5}$$

where w_i is obtained by dividing each of the 20 amino acid frequencies by the highest frequency, i.e., the most highly used amino acid will have its w_i = 1. For computing AAAI, we can use the copy number of tRNA genes as the reference set because most frequently used amino acids typically have more associated tRNA to carry them and because tRNA concentration is positively related to the copy number of tRNA genes, at least in several bacterial species and the yeast (Duret, 2000; Ikemura, 1992; Kanaya *et al.*, 1999; Percudani *et al.*, 1997).

It is important to keep in mind that CAI and AAAI are, respectively, gene-specific and peptide-specific, not codon-specific or amino acid-specific. It makes no sense to say that a codon has a CAI value of 0.75.

3.2 Problems with CAI and its current implementation

CAI has three major problems. Two problems are related to the compilation of the reference set of highly expressed genes, and one related to the computer implementation. Most published papers use the cai program in the EMBOSS (Rice *et al.*, 2000) distribution (typically referred to as the EMBOSS.cai program). I will consequently use EMBOSS.cai to illustrate implementation problems. Because the implementation problem is intertwined with the reference set, I will not try to separate the problems into reference-set-related and implementation-related.

3.2.1 Problem when w = 0

The first problem with CAI occurs when w = 0. It often happens that only a few genes are known to be highly expressed, even for model organisms such as the yeast (*Saccharomyces cerevisiae*). The number of codons one can compile from a small number of genes is consequently small, leading to some w values to be zero. For example, the frequently used codon usage table in the EMBOSS compilation Eyeastcai.cut for the budding yeast contains a number of zeros. In particular, in the CCN (coding for arginine) codon family, there are 43 CGT codons, but no CCG, CGA, or CGC codon.

The overuse of CGT and the avoidance of CCG, CGA and CGC codons in highly expressed genes make sense because the yeast genome contains six $tRNA^{Arg}$ genes all with anticodon ACG forming Watson-Crick base-pairing with the CGT codon, but no other $tRNA^{Arg}$ gene forming Watson-Crick base pairing with the other three CGN codons. The highly expressed genes included in the Eyeastcai.cut file apparently have strong codon usage bias favoring the CGT codon, taking advantage of the six ACG-tRNA^{Arg} genes to facilitate translation of arginine codons. While this illustrates well the codon-anticodon adaptation, it causes practical problems with computing CAI.

Given the 43 CGT codon and no other CGN codon in the reference set, the associated w value is therefore 1 for CGT but 0 for the other three. However, computing CAI requires taking the logarithm of w but there is no logarithm defined for w = 0. Different implementations of CAI typically would try to use some methods to avoid taking the logarithm of 0, but the resulting CAI can be outrageous. For example, if one uses the following sequence consisting of CGA, CGC, CGG codons only:

S = CGACGCCGGCGACGCCGGCGACGCCGGCGACGCCGG

as input to the EMBOSS.cai program (which is perhaps the most frequently used CAI calculator in practical research and available online at http://bioportal.cgb.indiana.edu/cgi-bin/emboss/cai), the resulting CAI value is 1 (the maximum CAI), which is totally unexpected and absolutely absurd.

We know that, among CGN codons, only CGT is represented in the reference set and all other three CGN codons have zero representation in the reference set. The sequence S consists of only CGA, CGC and CGG codon only but no CGT, and we therefore would expect the CAI to be at its minimum, i.e., 0. A CAI of 1 from EMBOSS.cai for sequence S is of course wrong. Unfortunately, given the scanty documentation of EMBOSS programs, one does not even know where to send bug reports without extensive searching through the internet. The implementation of CAI in DAMBE (Xia, 2001; Xia and Xie, 2001b) gives a CAI value of 0 for sequence S, with a note stating that there is insufficient information for computing CAI for the sequence.

Another way to avoid having w = 0 is to collect all sequenced protein-coding genes for a species and then choose those sequences with large CAI values as a reference set to compile a codon usage table. A compilation with all w values greater than 0 is available for the budding yeast in the EMBOSS compilation as Eysc_h.cut. The problem with this approach is that we are not sure if the included sequences in such a reference set are truly highly expressed.

3.2.2 Problems with codon families containing a single codon

EMBOSS.cai does not exclude codon families with a single codon in computing CAI. It is important to exclude such codons. Note that, for such codons (e.g., AUG and UGG in the standard genetic code), their corresponding w value will always be 1 regardless of codon usage bias of the gene. Although such codons will not contribute to the numerator in Eq. (9-4) because ln(1) = 0, they contribute to the denominator. If a gene happens to use a high proportion of methionine and tryptophan, then it will have a high CAI value even if the codon usage is not at all biased. Just add a string of ATG triplets to a sequence will substantially increase its CAI.

Because EMBOSS.cai does not exclude codon families with a single codon, one should be cautious in interpreting results from it. For example, if the input sequence consists of multiple ATG codons, such as

S = ATGATGATG……

then the EMBOSS.cai program will yield a CAI value of 1, based on the web interface of EMBOSS.cai available at http://bioportal.cgb.indiana.edu/cgi-bin/emboss/cai. Of course such a CAI value is wrong. A correctly computed CAI value should not depend on the frequencies of methionine and tryptophan. The implementation of CAI in DAMBE (Xia, 2001; Xia and Xie, 2001b) gives a CAI value of 0 for sequence S, with a note stating that there is insufficient information for computing CAI for the sequence.

3.2.3 Problems with amino acids coded by two different codon families

EMBOSS.cai also produce other perplexing output. Suppose we now use a sequence consisting entirely of CGT codons and expect the resulting CAI to be 1 by using the Eyeastcai.cut reference set (Recall that the reference set contains 43 CGT codons but no CGA, CGC or CGG codon). The resulting CAI value from the EMBOSS.cai program is 0.140 instead of 1. This is again unexpected. It turns out that amino acid arginine is coded by two codon families, the CGN codon family we have mentioned, and the AGR codon family. The largest codon frequency among these six codons is 314 (for AGA codon). So the w value for CGT is not 1 (43/43) as we have thought, but is only 0.1369 (= 43/314). For standard genetic code, there are three amino acids (arginine, leucine and serine) each coded by two different codon families. EMBOSS.cai does not separate the two codon families for each amino acid, but treated them as three six-member codon families. This is not appropriate because the codon usage bias in one codon family (e.g., the CGN codon family) translated by one set of tRNAs is much obscured by the codon usage in another codon family (e.g., the AGR codon family) translated by another set of tRNA genes.

The implementation of CAI in DAMBE (Xia, 2001; Xia and Xie, 2001b) separates the “six-codon family” into two separate codon families, with one family containing two codons and another containing four. For example, one arginine codon family contains the two AGR codons and the other contains the four CGN codons.

3.2.4 Problems with initiation and termination codons

Strictly speaking, CAI is an index measuring the efficiency of translation elongation. So its calculation should not include initiation and termination codons because these special codons are more related to translation initiation and termination than translation elongation. EMBOSS.cai does not exclude the termination codons. However, because each protein-coding gene is generally expected to contain only one termination codon, the effect of including the termination codon in computing CAI is small with long gene sequences.

The implementation of CAI in DAMBE (Xia, 2001; Xia and Xie, 2001b) does not exclude the initiation codon in prokaryotic genomes, but excludes the termination codon in computing CAI.

3.2.5 The problem with the compilation of the reference set of genes

Early reference sets of genes include known highly expressed genes such as genes coding for ribosomal proteins. An average *E. coli* cell in the exponential growing phase contains about 15,000 ribosomes made of two subunits. The large subunit contains the 5S rRNA (120 bases long) and 23S rRNA (about 2900 bases long, e.g., *E. coli* K12 strain has seven 23S rRNA genes, with six being 2904 bases long and one being 2905 bases long), together with 31 different proteins. The small subunit contains the 16S rRNA (1542 bases long) and 21 different proteins. Nearly all ribosomal proteins are present in single copies in each ribosome. This implies that ribosomal proteins are all highly expressed and exist in about 15,000 copies per cell.

The first large-scale compilation of reference sets is derived from TransTerm (Brown *et al.*, 1994; Dalphin *et al.*, 1996), which also outputs CAI values for genes from species with a reference set of highly expressed genes. Such genes, together with associated parameters such as CAI, are compiled for each species in a file named ****.dat, where '****' is usually a four letter code made from the organism's genus and species. For example, the codon usage table for *Homo sapiens* is Hsap.dat. The subset of genes with the highest CAI values are found in the file named ****_H.dat. TransTerm also outputs these files in GCG format (Dalphin *et al.*, 1996), named as ****.cod. Note that gene sequences in the ****_H.dat files are not necessarily highly expressed genes because their expression is not verified by gene expression studies.

These ****.dat files can be formatted as codon frequency tables that can be used as the reference set of genes for computing CAI values. The first large-scale distribution of the reformatted codon frequency tables came with the release of EMBOSS (Rice *et al.*, 2000). The EMBOSS-reformatted codon frequency tables are stored in files named E*.cut where the prefix E is presumably for EMBOSS and the file type "cut" is for codon usage table. The * part in the file is typically a species designation, but unfortunately is not standardized. Because EMBOSS is open-source, and consequently because everybody can contribute to it with little restriction, there was an undesirable proliferation of E*.cut files. For example, you will find Ehum.cut, Ehuman.cut, Eeco.cut, Eeco_h.cut, Eecoli.cut, Emus.cut, etc. In some cases, the species is easy to tell. For example, the first two file names in the previous sentence refer to human, the next three refer to *Escherichia coli,* with the middle one referring to highly expressed *E. coli* genes (i.e., from genes with high CAI values, not from genes experimentally verified to be highly expressed), and the last refers to *Mus musculus.* Names ending with cp refer to chloroplast genes. For example, Emzecp.cut, is from maize chloroplast genes. File names ending with mt are from mitochondrial genes. For example, Eyscmt.cut is derived from the yeast (*Saccharomyces cerevisiae*) mitochondrial genes.

There are two major problems with the EMBOSS compilation of reference genes. First, the nonstandard species designation, coupled with a lack of documentation typically associated with open-source software, has led to a profound confusion as to which file refers to which species, who compiled the reference codon usage table and how the table is obtained. Second, the reference set of genes are supposed to be highly expressed, but it is difficult to define highly expressed genes in multicellular eukaryotes because a gene may be highly expressed only in a certain tissue at a certain time.

I present below two alternatives to CAI computation. One is a minor revision of CAI computation by using a codon frequency table derived from anticodons of all tRNA genes in a genome. The rest of the computation is the same as the conventional CAI. The other is an entirely different measure of codon usage bias, but also based on the distribution of tRNA anticodons.

4. INDICES OF CODON-ANTICODON ADAPTATION

One may think that, because efficiency of translation elongation depends much on whether the codon usage maximizes the use of codons corresponding to the most abundant tRNA species (Bennetzen and Hall, 1982; Gouy and Gautier, 1982; Ikemura, 1981, 1982; Ikemura, 1992; Xia, 1998b), we can use tRNA relative abundance as the basis for a reference set. It is less controversial to define translationally optimal codons as those matching the anticodon of most abundant tRNA species than as those matching the most frequent codons in a subset of highly expressed genes because the latter may not be representative of highly expressed genes.

From the frequencies of tRNA anticodons, one can obtain the frequencies of codons that form Watson-Crick base pairs with the anticodons and use such a codon frequency table as a reference set for computing CAI. Such a reference set, with all the w values based on the tRNA-derived "codon frequencies", is presented in Tables 9-2 and 9-3.

Table 9-2. tRNAs translating two-fold codon families from *Saccharomyces cerevisiae.*

AA(1)	Codon(2)	T(3)	w(4)	F(5)
Arg	AGA	11	1	314
Arg	AGG	1	0.091	1
Asn	AAC	10	1	208
Asn	AAU	0	0	11
Asp	GAC	16	1	202
Asp	GAU	0	0	112
Cys	UGC	4	1	3
Cys	UGU	0	0	39
Gln	CAA	9	1	153
Gln	CAG	1	0.111	1
Glu	GAA	14	1	305
Glu	GAG	2	0.143	5
His	CAC	7	1	102
His	CAU	0	0	25
Leu	UUA	7	0.7	42
Leu	UUG	10	1	359
Lys	AAA	7	0.5	65
Lys	AAG	14	1	483
Phe	UUC	10	1	168
Phe	UUU	0	0	19
Ser	AGC	2	1	6
Ser	AGU	0	0	4
Tyr	UAC	8	1	141
Tyr	UAU	0	0	10

(1) Amion acid carried by tRNA

(2) Codons forming Watson-Crick base pair with the anticodon of tRNA

(3) Copy number of tRNA gene.

(4) w_i equals T_i divided by the maximum T within each codon family

(5) the codon frequencies of highly expressed yeast protein-coding genes compiled in the Eyeastcai.cut file distributed with EMBOSS (Rice et al., 2000).

The idea of using the relative frequencies of tRNA anticodons to define translationally optimal codons is not new. Both the frequency of optimal codons, or F_{op} (Ikemura, 1985) and the codon bias index, or CBI (Bennetzen and Hall, 1982) incorporate the relative frequencies of tRNA anticodons as a reference to identify translationally optimal codons.

Table 9-3. tRNA data in the genome of the budding yeast, *Saccharomyces cerevisiae.* Only four-fold codon families are included. Symbols as in Table 9-2.

AA	Codon	T	w	F
Ala	GCA	5	0.455	6
Ala	GCG	0	0	0
Ala	GCC	0	0	130
Ala	GCU	11	1	411
Arg	CGA	0	0	0
Arg	CGG	1	0.167	0
Arg	CGC	0	0	0
Arg	CGU	6	1	43
Gly	GGA	3	0.188	1
Gly	GGG	2	0.125	2
Gly	GGC	16	1	9
Gly	GGU	0	0	459
Ile	AUA	2	0.154	0
Ile	AUC	0	0	181
Ile	AUU	13	1	149
Leu	CUA	3	1	14
Leu	CUG	0	0	1
Leu	CUC	1	0.333	1
Leu	CUU	0	0	2
Pro	CCA	10	1	211
Pro	CCG	0	0	0
Pro	CCC	0	0	2
Pro	CCU	2	0.2	10
Ser	UCA	3	0.273	7
Ser	UCG	1	0.091	1
Ser	UCC	0	0	133
Ser	UCU	11	1	192
Thr	ACA	4	0.364	2
Thr	ACG	1	0.091	1
Thr	ACC	0	0	164
Thr	ACU	11	1	151
Val	GUA	2	0.143	0
Val	GUG	2	0.143	5
Val	GUC	0	0	231
Val	GUU	14	1	278

The approach of using relative tRNA abundance as the reference is particularly attractive given the high correlation between relative tRNA abundance and the copy number of tRNA genes (Duret, 2000; Ikemura, 1992; Kanaya *et al.*, 1999; Percudani *et al.*, 1997). The availability of many genomes as well as the ease of identifying tRNA genes (Lowe and Eddy, 1997) allow us to quickly obtain all tRNA genes in a genome, identify their anticodons and the codons that form Watson-Crick base pair with these anticodons. These codons can then form a "codon usage" table and used as a reference set for computing CAI.

In general, the most frequently used codons in each codon family (last column in Table 9-2 and Table 9-3) correspond to the most abundant tRNA species translating that codon family, although there are exceptions. We will look at these exceptions in more detail later.

4.1 CAI with a tRNA anticodon-derived codon usage table

It is simple to replace the reference set in CAI computation by the codon usage table derived from tRNA anticodons. For any particular species, what we need to do is to compile the copy number of each tRNA genes, identify the anticodon of each tRNA gene, and obtain a frequency table of their anticodons. This table of anticodon frequency can be directly translated into a table of codons that form Watson-Crick base-pairing with these anticodons. A codon frequency table obtained in this way can then be used as a reference set to compute CAI, referred hereafter as tCAI to distinguish it from the conventional CAI. This approach has been made easy by the proliferation of genomic sequencing projects and the tRNA scanning software (Lowe and Eddy, 1997).

Tables 9-2 and 9-3 together show such a codon usage table for the budding yeast, *Saccharomyces cerevisiae,* together with calculated w values for computing tCAI. The resulting tCAI for yeast protein-coding genes and the conventional CAI based on EMBOSS compilation Eysc_h.cut are highly correlated, with r = 0.98 (Figure 9-1). This suggests the potential of using tRNA anticodons as a reference set for computing codon adaptation index.

The ultimate test of an index of translation elongation efficiency is on whether it can, together with the relative mRNA concentration, predict protein production. CAI has been used for this purpose (Futcher *et al.*, 1999; Gygi *et al.*, 1999). When the data in these two papers are used, tCAI is very slightly (but not significantly) better in predicting protein production than CAI (details in the last section of the chapter).

The high correlation between the tRNA-based CAI and the conventional CAI may lead one to conclude that the former is a highly satisfactory replacement for the latter, without any evils associated with the poorly defined and poorly documented reference sets associated with the latter. Unfortunately, this is not true.

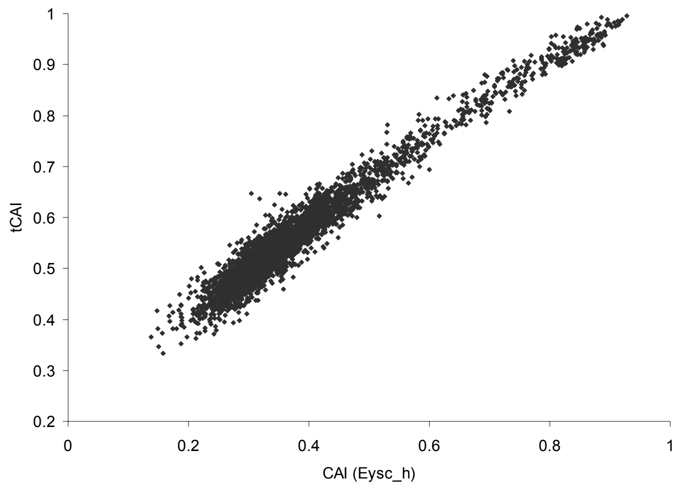


Figure 9-1. CAI based on tRNA anticodons of the yeast (*Saccharomyces cerevisiae*), designated tCAI, and conventional CAI based on EMBOSS compilation of Eysc_h.cut. Computed by ignoring all codons with their corresponding w values equal to 0 (because no logarithm is defined for 0).

There is a serious problem that limits the utility and interpretation of tCAI . For some species, the number of tRNA genes is relatively small, with the consequence that some w values are zero (Tables 9-2 and 9-3). These w values typically correspond to codons that are rarely used in highly expressed genes, but more frequent in lowly expressed genes. If we ignore all codons with their corresponding w values equal to 0, then the resulting CAI values for highly expressed genes will be little affected, but CAI values for lowly expressed genes tend to be elevated. This is already visible in Figure 9-1 which compares the conventional CAI based on the reference codon usage table Eysc_h.cut and the tRNA-based CAI. The effect is more obvious with *Escherichia coli* with fewer tRNA genes (Figure 9-2).

The seriousness of the problem is not well illustrated with the yeast or *E. coli* data because they are not the species with the fewest tRNA genes. For example, *Mycoplasma genitalium* G37 genome (NC_000908) contains only 38 tRNA genes, and *M. pulmonis* genome (NC_002771) contains only 29 tRNA genes (of which only 28 may be functional because one of the two $tRNA^{Trp}$ genes, MYPU_TRNA_TRP_1, does not form the anticodon loop properly, i.e., it does not have the seven-nucleotide anticodon loop formed by the anticodon flanked by two nucleotides and held by a stem). Such a small number of tRNA genes will result in about half of the w values being zero. The resulting w values would generate a very high CAI value for every gene (i.e., most genes will have CAI = 1) if we ignore all codons with their corresponding w values equal to zero. Therefore, while the yeast result (Figure 9-1) suggests the potential of using tRNA genes to compute CAI, it is necessary to develop an alternative index of codon usage bias.

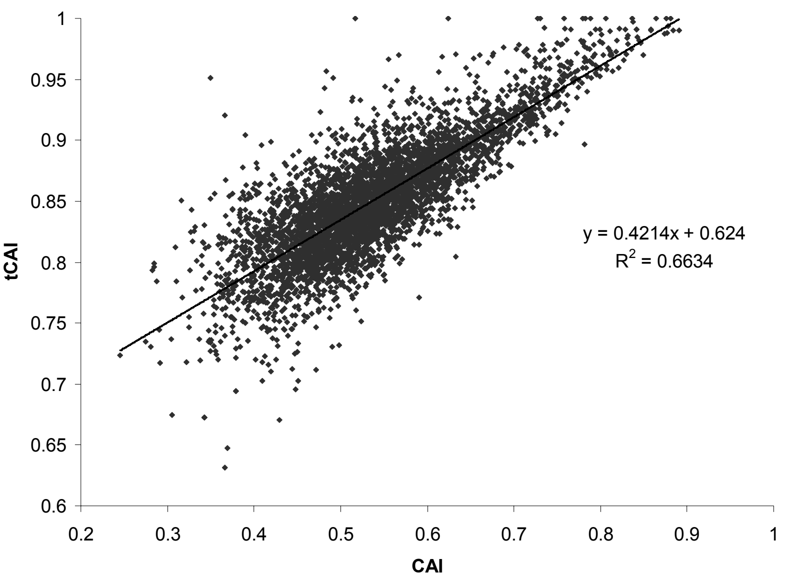


Figure 9-2. Relationship between tCAI and conventional CAI (based on EMBOSS compilation of Eeco_h.cut) for *Escherichia coli*. Computed by ignoring all codons with their corresponding w values equal to 0 (because no logarithm is defined for 0).

4.2 Codon-anticodon adaptation index (CAAI)

Before we start developing our new index, it is important to review a few exceptional codon families in Table 9-2 and Table 9-3. We have previously asked why indices based on tRNA availability, indirectly measured by the number of tNRA gene copies because the two are generally positively correlated in several bacterial species and the budding yeast (Duret, 2000; Ikemura, 1992; Kanaya *et al.*, 1999; Percudani *et al.*, 1997), often do not perform as well as CAI which ignores tRNA but is based entirely on the codon usage of highly expressed genes. Examining the exceptional codon families in Table 9-2 and Table 9-3 will help answer the question.

Table 9-2 and Table 9-3 show that codon usage of highly expressed genes does not always match the tRNA gene with the largest number of gene copies in the genome. Although the codon that forms perfect base-pairing with the tRNA anticodon is preferentially used within each codon family in most cases, there is an obvious exception involving cysteine (Table 9-2).There are four tRNA^{Cys} genes with the anticodon matching codon UGC but no tRNA^{Cys} with an anticodon matching codon UGU. We would have expected UGC codon to be preferentially used over the UGU codon. However, the opposite is true (Table 9-2). While there has been no satisfactory explanation for highly expressed genes to prefer UGU codon against UGC codon, the observation that codon usage of highly expressed genes do not always match the tRNA gene with the largest number of gene copies in the genome may explain why indices based on relative tRNA abundance may not perform as well as indices based on the codon frequency of known highly expressed genes.

Similar exceptions are also present in the four-fold codon families (Table 9-3). For example, there is no tRNA^{Gly} with an anticodon matching perfectly the GGU codon, yet the codon is by far the most frequently used in highly expressed protein-coding genes (Table 9-3). Another exception is tRNA^{Thr}. There is no tRNA^{Thr} with an anticodon that forms Watson-Crick base pair with codon ACC, yet ACC is the most frequently used codon in the codon family (Table 9-3). In particular, while the exceptional case in Table 9-2 involves an infrequently used amino acid (cysteine), the exceptional cases in Table 9-3 involve relatively frequently used amino acids (glycine and threonine). It is difficult to argue that natural selection favouring an increased efficiency of translation for highly expressed genes should turn a blind eye on glycine and threonine codon families while imparting a strong effect on other codon families.

The exceptional cases are not unique in the yeast, but can also be found in many other species (although not always the same codon families being exceptional). Because of these exceptions, we cannot always be sure of which codon is translationally optimal. For example, based on tRNA copy numbers, we would have predicted codons UGC, GGC, and ACU to be translationally optimal in the UGY, GGN and ACN codon families, respectively. However, based on the codon frequencies of known highly expressed genes, we expect codons UGU, GGU and ACC to be translationally optimal in the UGY, GGN and CAN codon families, respectively. This should be kept in mind when we learn indices of codon usage bias such as RSCU and CAI.

Readers who still remember the codon-anticodon adaptation hypothesis and the wobble versatility hypothesis that we encountered in Chapter may have noticed these two hypotheses, specifically formulated for vertebrate mitochondrial genomes, are not applicable for eukaryotic nuclear genomes. We have already mentioned exceptions to the codon-anticodon adaptation hypothesis (i.e., the cysteine codon family in Table 9-2, and glycine and threonine codon families in Table 9-3). The wobble versatility hypothesis is also not very useful in predicting the wobble nucleotide in eukaryotic tRNA encoded by nuclear genomes. For example, the wobble versatility hypothesis states that tRNA translating Y-ending codons should have nucleotide G at the wobble site (the first nucleotide of the anticodon) because G can pair with both C and U. However, however, most tRNAs translating Y-ending codons within the four-fold degenerate codon families have nucleotide A (Table 9-3).

Because the exceptional cysteine codon family in Table 9-2 involves an under-used amino acid (cysteine accounts for about 1% in protein-coding genes in the genome of the budding yeast, *Saccharomyces cerevisiae* and 2.2% in the 890 protein-coding sequences in human chromosome 22), its effect on the codon usage index is small. More serious problems are present in Table 9-3 because exceptions involve frequently used amino acids such as glycine and threonine. If we use relative tRNA copy number to generate w (Tables 9-2 and 9-3), and if the input sequence happens to contain many GGU (glycine codon) and ACC (threonine codon), then the input sequence will have a rather small CAI value, although the biased usage of these codons is characteristic of highly expressed genes (Table 9-3).

We note in Table 9-3 that each four-fold codon family is generally translated by at least two types of tRNAs, one with a wobble U at the tRNA anticodon to translate R-ending codons and the other with a wobble G at the tRNA anticodon to translate Y-ending codons (where R stands for A or G and Y stand for C or T/U). As we have briefly explored in Chapter 8 on the cost of wobble pairing, there might be little cost in wobble-translating G-ending codons given a wobble U at the tRNA anticodon. So the wobble U may not imply any selection against G-ending codons. Similarly, there might be little selection against U-ending codons when the wobble site of the tRNA anticodon is G. Thus, we may simply collapse the four-fold codon families into "two-fold" codon families with R-ending and Y-ending codons.

Table 9-4 results from collapsing Table 9-3 into R-ending and Y-ending codons. This has three advantages. First, it eliminates the exceptional codon families that do not show the nice association between tRNA gene copy number and codon frequencies of highly expressed genes. Second, it has essentially eliminated all zero w values. Third, it reduces codon usage bias not associated with selection for optimizing codon-anticodon adaptation. For example, in extremely AT-biased genome maintained by strong AT-biased mutation spectrum, a four-fold codon family may have many A-ending and T-ending codons but few C-ending or G-ending codons. Such mutation-mediated codon usage bias has little to do with maximizing translation elongation rate and should not confound the computation of an index such as CAI intended to measure efficiency of translation elongation. By collapsing such codons into R-ending and Y-ending codons, such mutation-mediated codon usage bias is eliminated or at least substantially reduced.

Table 9-4. Collapsing Table 9-3 to R-ending and Y-ending codons. Symbols as in Table 9-2.

AA	Codon	T	w	F
Ala	GCR	5	0.455	6
Ala	GCY	11	1	541
Arg	CGR	1	0.167	0
Arg	CGY	6	1	43
Gly	GGR	5	0.313	3
Gly	GGY	16	1	468
Ile	AUR	2	0.154	0
Ile	AUY	13	1	330
Leu	CUR	3	1	15
Leu	CUY	1	0.333	3
Pro	CCR	10	1	211
Pro	CCY	2	0.2	12
Ser	UCR	4	0.364	8
Ser	UCY	11	1	325
Thr	ACR	5	0.455	3
Thr	ACY	11	1	315
Val	GUR	4	0.286	5
Val	GUY	14	1	509

We now again have two roads diverged in the yellow wood. One is to continue to compute CAI, by using a reference codon usage table with four-fold codon families collapsed as in Table 9-4. Now for each input sequence for which we need to compute the CAI value, we count the codon frequencies in the two-fold families and use the w values in Table 9-2 but, for four-fold codon families, we count the frequencies of all R-ending and Y-ending codons and use the w values in Table 9-4. This allows us to use tRNA-derived codon frequencies for any species. The only disadvantage is that there are still some w values being zero, and we would always feel guilty for not using all the information in the data.

The other road we can take is to develop an entirely new index. Note that all codon families in Tables 9-2 and 9-4 are effectively "two-fold" after collapsing the four-fold codon families into the R-ending and Y-ending codons. Designating the codon frequencies of such a "two-fold" codon family i as N_{i1} and N_{i2}, the deviation of N_{i1} and N_{i2} from equal codon usage can be measured by:

$$X_i^2 = \frac{(N_{i1} - E_i)^2}{E_i} + \frac{(N_{i2} - E_i)^2}{E_i} = \frac{(N_{i1} - N_{i2})^2}{M_i} \tag{9.6}$$

where $M_i = (N_{i1} + N_{i2})$, and $E_i = M_i / 2$.

Summing up the X_i^2 values would generate an overall measure of codon usage bias for the gene. However, there are two problems. First, X_i^2 is dependent on the number of codons in codon family i so that a codon family with a large (N_{i1} + N_{i2}) tends to have a larger X_i^2. To eliminate the dependence we need to divide the summation of X_i^2 by the summation of M_i. Second, we have not yet incorporated the information of the reference set. This we can do by introducing a sign function for codon family i:

$$S_i = sign[(N_{i1} - N_{i2})(T_{i1} - T_{i2})] \tag{9.7}$$

where T_{i1} and T_{i2} stand for the frequencies of the two tRNA-derived codons, i.e., the column headed by T in Tables 9-2 and 9-4. The sign function is positive when the codon usage bias is in the same direction as the tRNA bias, and negative when it is not. These considerations now lead to a new index called codon-anticodon adaptation index (CAAI, following the wisdom that two A's are better than one):

$$CAAI = \frac{\sum_{i=1}^{n}\left(S_i X_i^2\right)}{\sum_{i}^{n} M_i} \tag{9.8}$$

where S_i determines whether X_i^2 is to be positive or negative. Obviously, if codon usage is biased to favor the codon with the fewest cognitive tRNA species, then the contribution of the codon usage bias should be negative.

The value of CAAI ranges from -1 to 1 and can be interpreted in the same way as a correlation coefficient. The relationship between CAAI for *Escherichia coli* K12 (NC_000913) and the conventional CAI computed with the EMBOSS compilation Eeco_h.cut is stronger (Figure 9-3), with R^2 = 0.7294, than that between tCAI and CAI (Figure 9-2), with R^2 = 0.6634. Thus, it seems that CAAI is a better replacement of CAI than tCAI. The relationship in Figure 9-3 also appears more linear and more normally distributed than that in Figure 9-2. This makes it easy to re-scale CAAI to be within the range of 0 and 1 as CAI. The linear relationship (Figure 9-3) implies that any analysis involving CAI can also be performed with CAAI with no further data transformation.

A more objective check of the relative value of CAAI and CAI is to see which one is better associated with the translation initiation signal, based on the assumption that a highly expressed protein should not only have strong codon usage bias to speed up translation elongation, but also a strong initiation signal to increase initiation efficiency. For prokaryotes, the initiation site is located by the binding of Shine-Dalgarno (SD) sequence of small subunit rRNA to the anti-SD sequence on the 5'-end of mRNA upstream of the initiation codon (Shine and Dalgarno, 1974, 1975a). The efficiency of translation initiation depends, in a non-linear fashion, on the binding strength (S) and the distance of the binding to the initiation codon (D). One can obtain a set of known highly expressed proteins in *E. coli*, characterize S and D, and check to see if CAI or CAAI is a better predictor of features of S and D in highly expressed genes.

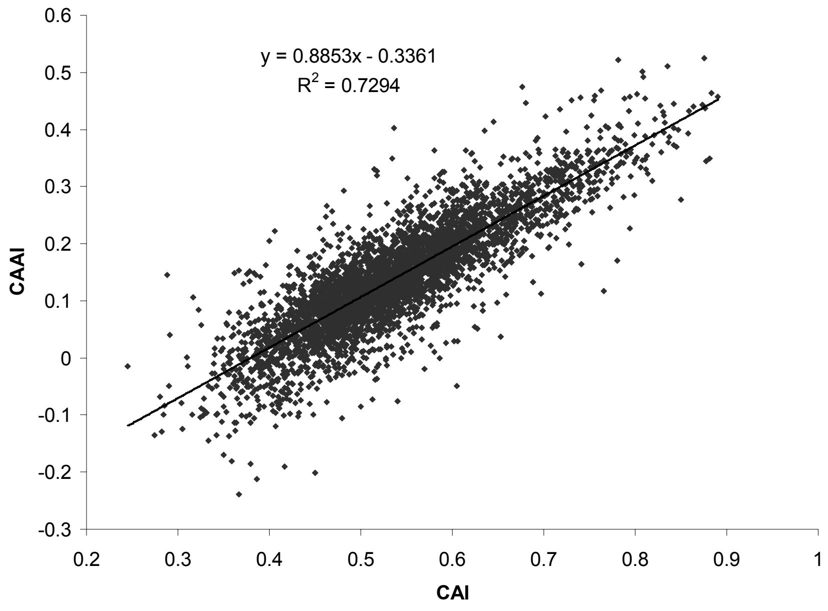


Figure 9-3. Relationship between CAAI and conventional CAI (based on EMBOSS compilation of Eeco_h.cut) for *Escherichia coli*. Results from DAMBE(Xia, 2001; Xia and Xie, 2001b)

5. WHY CAI OR CAAI SHOULD NOT BE TAKEN AS A MEASURE OF GENE EXPRESSION?

There is a great deal of confusion concerning the relationship between CAI (or CAAI) and gene expression, especially when gene expression refers to the expression level of mRNA. As I have emphasized before, CAI and CAAI are intended to measure the efficiency of translation elongation. When there are selection favoring an increased production of a certain protein, then the selection can act at both the translation level and transcription level, so that the protein-coding gene may both be transcriptionally and translationally highly expressed. For this reason, an index such as CAI or CAAI that increases with translation efficiency will also be spuriously increased with transcription efficiency.

There is no theoretical basis that CAI or CAAI should always be positively correlated with mRNA level. In fact, it is easy to think of scenarios when CAI or CAAI would be negatively correlated with mRNA level. Let us first learn a fact about CAI or CAAI that might at first erode your confidence in such indices. The fact is that these indices depend on the AT% (or GC% sometimes referred to as GC content) of the gene, with AT-rich genes having low CAI or CAAI values (Figure 9-4).

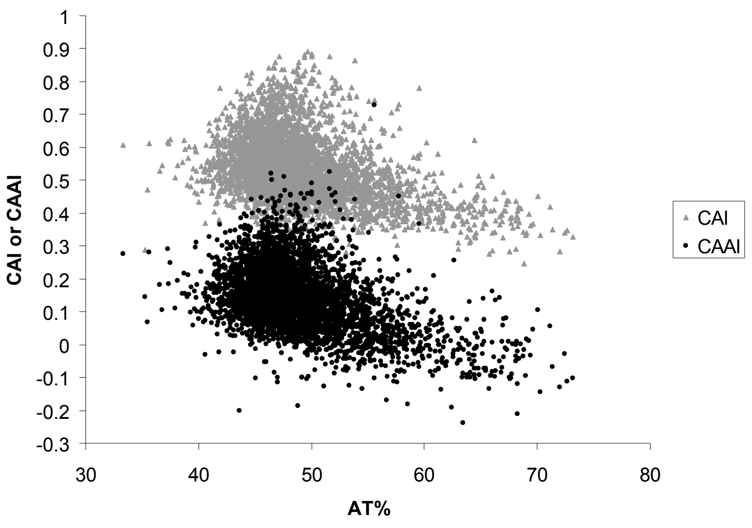


Figure 9-4. Dependence of CAI or CAAI on AT% of the gene, based on *Escherichia coli* K12 data (NC_000913).

This dependence of CAI and CAAI of a gene on the AT% of the gene may at first appear disconcerting, but is in fact quite natural given the fact that CAI and CAAI essentially measure codon-anticodon adaptation. Consider a two-fold codon family ending with either C or U. Such a codon family is typically translated by a tRNA with a G at the wobble site, because G can pair with both C and U. Given the wobble G in the anticodon, we would expect C-ending codons to be maximally used. However, for an AT-rich gene, the codon usage is almost necessarily dominated by U in this Y-ending codon family. So we have poor codon-anticodon adaptation in these

AT-rich genes. This would contribute to low CAI and CAAI values associated with high AT% as we can observe in Figure 9-4.

Is this explanation correct or sufficient? At this point a smart student will typically raise a strong objection as follows. Instead of considering only the Y-ending codons in the previous paragraph, we consider both R-ending and Y-ending codons, assuming that the former is translated by tRNA with a U at the anticodon wobble site and the latter by tRNA with a G at the anticodon wobble site. This is summarized in Figure 9-5.

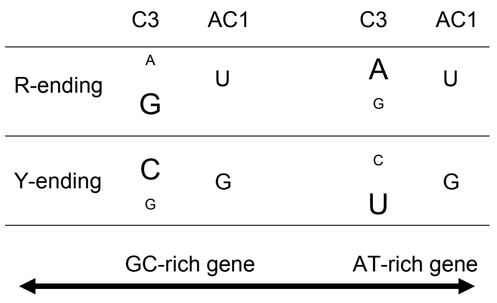


Figure 9-5. Effect of increasing and decreasing AT% on CAI and CAAI, assuming that R-ending codons are translated by tRNA with a U at its anticodon wobble site and Y-ending codons by tRNA with a G at its anticodon wobble site. A small "A" and a large "G" mean the R-ending codon family contains few A-ending codons but many G-ending codons. C3 – nucleotide at the third codon position; AC1 – nucleotide at the first anticodon position.

We focus first on the right half of Figure 9-5 with AT-rich genes in which R-ending codon families are dominated by A-ending codons and Y-ending codon families are dominated by U-ending codons (Figure 9-5, right half). While the Y-ending codon families of these genes will be biased to have many U-ending codons and few C-ending codons, resulting in fewer Watson-Crick C/G pairs and more wobble U/G pairs during translation, the AT-richness also implies more A-ending codons and fewer G-ending codons in R-ending codon families, resulting in more Watson-Crick A/U pairs and fewer G/U wobble pairs during translation. The latter should contribute to an increased CAI or CAAI and offset the CAI-decreasing effect of Y-ending codon families. Moreover, for GC-rich genes (the left part of Figure 9-5), Y-ending codon families should improve their contribution to CAI (or CAAI) because these codon families now should feature many C-ending codons and few U ending codons. In contrast, R-ending codons should now contribute poorly to CAI (or CAAI) because they now have many G-ending codons that need to be wobbly translated (Figure 9-5). So the argument in the previous paragraph does not seem logical or sufficient to explain the negative correlation between CAI (and CAAI) and AT%.

The answer to the objection turns out to be quite simple. While Y-ending codons are typically translated by one type of tRNA with a wobble G at its anticodon, R-ending codons typically have two types of tRNA, one with a U at the anticodon wobble site to translate A-ending codons and the other with a C at the anticodon wobble site to translate G-ending codons. This is summarized in Figure 9-6. In short, it is the Y-ending codon families whose contribution to CAI or CAAI is sensitive to the AT% of the gene. The contribution of R-ending codon families may be largely ignored. Focusing on the Y-ending codon families only, we can see easily that genes with high AT% should have few C/G pairs and many wobble U/G pairs during translation. This explains well the negative relationship between gene AT% and CAI or CAAI.

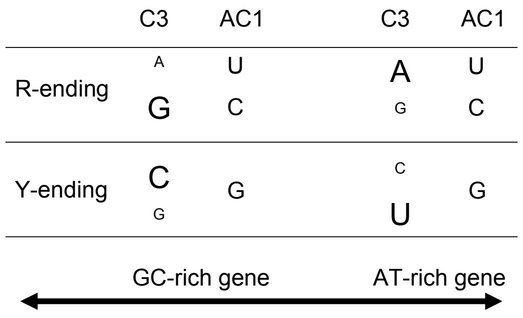


Figure 9-6. A more realistic display of the effect of increasing and decreasing AT% on CAI and CAAI. Symbols mean the same as in Figure 9-5.

For readers who want to see real data instead of a graphic abstraction, I list the human tRNA distribution in Table 9-5.

Table 9-5. Frequency distribution of tRNA genes in human.

AA(1)	Codon	tRNA with anticodon	N_{tRNA}(2)
Ala	GCA	TGC	9
Ala	GCG	CGC	5
Ala	GCC	GGC	0
Ala	GCU	AGC	29
Arg	AGA	TCT	6
Arg	AGG	CCT	5
Arg	CGA	TCG	6
Arg	CGG	CCG	5
Arg	CGC	GCG	0
Arg	CGU	ACG	7
Asn	AAC	GTT	28
Asn	AAU	ATT	1
Asp	GAC	GTC	18
Asp	GAU	ATC	0

AA[1]	Codon	tRNA with anticodon	N_{tRNA}[2]
Cys	UGC	GCA	30
Cys	UGU	ACA	0
Gln	CAA	TTG	11
Gln	CAG	CTG	21
Glu	GAA	TTC	12
Glu	GAG	CTC	13
Gly	GGA	TCC	9
Gly	GGG	CCC	7
Gly	GGC	GCC	15
Gly	GGU	ACC	0
His	CAC	GTG	11
His	CAU	ATG	0
Ile	AUA	TAT	5
Ile	AUC	GAT	5
Ile	AUU	AAT	14
Leu	UUG	CAA	6
Leu	UUA	TAA	7
Leu	CUA	TAG	3
Leu	CUG	CAG	10
Leu	CUC	GAG	0
Leu	CUU	AAG	12
Lys	AAA	TTT	17
Lys	AAG	CTT	17
Phe	UUC	GAA	12
Phe	UUU	AAA	0
Pro	CCA	TGG	7
Pro	CCG	CGG	4
Pro	CCC	GGG	0
Pro	CCU	AGG	10
Ser	UCA	TGA	5
Ser	UCG	CGA	4
Ser	UCC	GGA	0
Ser	UCU	AGA	11
Ser	AGC	GCT	8
Ser	AGU	ACT	0
Thr	ACA	TGT	8
Thr	ACG	CGT	6
Thr	ACC	GGT	0
Thr	ACU	AGT	10
Tyr	UAC	GTA	14
Tyr	UAU	ATA	1
Val	GUA	TAC	5
Val	GUG	CAC	16
Val	GUC	GAC	0
Val	GUU	AAC	11

(1) amino acid

(2) Number of tRNA gene copies.

In general, Y-ending codons are translated by one type of tRNA with a wobble G at its anticodon, while R-ending codons typically have two types of tRNA, one with a U at the anticodon wobble site to translate A-ending codons and the other with a C at the anticodon wobble site to translate G-ending codons.

At this point a smart student may again argue that Table 9-5 is irrelevant. The dependence of CAI and CAAI on AT% is demonstrated for *E. coli* (Figure 9-4), but the tRNA data in Table 9-5 is for human. The human data would be relevant if we have shown a negative correlation between CAI (or CAAI) and AT% in human genes but, to explain the negative correlation between CAI (or CAAI) and AT% in *E. coli* genes (Figure 9-4), don't we need tRNA data from *E. coli*?

This is an excellent question. Some people, in response to this question, will quote the dogmatic assertion that "what is true for *E. coli* is also true for the cow, only truer". This is not the right way of carrying out science. The proper response is to present tRNA data for *E. coli* to demonstrate the same pattern seen in human tRNA data (Table 9-6 and Table 9-7).

Table 9-6 shows the frequency distribution of tRNA genes in *E. coli* that translate Y-ending codons in each codon family. These codons are translated by one type of tRNA with a wobble G at its anticodon. The anticodon wobble site of almost all these tRNA genes is occupied by a G, with tRNA^{Arg} being the only exception. This is consistent with the wobble versatility hypothesis (Xia, 2005c) stating that the wobble site of tRNA anticodons is determined by the necessity of wobble pairing. Given the anticodon wobble nucleotide being G, an increase in U-ending codons and a decrease of C-ending codon as a consequence of AT-richness in the gene naturally will lead to a decrease in codon-anticodon adaptation, i.e., more U/G wobble pairing and fewer C/G pairing during translation.

In contrast to Y-ending codons that are typically translated by tRNA with a wobble G at its anticodon wobble site, R-ending codons in *E. coli,* similar to those in human, are typically translated by two types of tRNAs, one with a U and the other with a C at the anticodon wobble site (Table 9-7). This is similar to human tRNA (Table 9-5). Therefore, an increase in GC% in a gene has relatively little effect on its codon-anticodon adaptation for R-ending codons.

To summarize, Y-ending codons are typically translated by one type of tRNA with a wobble G at its wobble site, and R-ending codons are typically translated by two types of tRNA, one with a C and another with a U at its wobble site. This is generally true from *E. coli* to human. When AT% of the gene increases, resulting in many U-ending codons in Y-ending codon families, the increased U/G wobbling will decrease the CAI value of those AT-rich genes. The increased AT% would also increase A-ending codons and decrease G-ending codons, but such increases in AT% has little effect on codon-anticodon adaptation because R-ending codons generally have two types of tRNA for translation. When GC% increases, or in the extreme case when all Y-ending codons are C-ending codons, the resulting perfect C/G pairing increases the CAI value. This explains the negative relationship between CAI (or CAAI) and AT% (Figure 9-4).

Table 9-6. Frequency distribution of tRNA genes for translating Y-ending codons in *E. coli*. Y-ending codons in each codon family are typically translated by tRNA with a wobble G at the anticodon. Symbols as in Table 9-5.

AA	Codon	tRNA with anticodon	N_{tRNA}
A	GCC	GGC	2
A	GCU	AGC	0
C	UGC	GCA	1
C	UGU	ACA	0
D	GAC	GUC	3
D	GAU	AUC	0
F	UUC	GAA	2
F	UUU	AAA	0
G	GGC	GCC	4
G	GGU	ACC	0
H	CAC	GUG	1
H	CAU	AUG	0
I	AUC	GAU	3
I	AUU	AAU	0
L	CUC	GAG	1
L	CUU	AAG	0
N	AAC	GUU	4
N	AAU	AUU	0
P	CCC	GGG	1
P	CCU	AGG	0
R	CGC	GCG	0
R	CGU	ACG	4
S	AGC	GCU	1
S	AGU	ACU	0
S	UCC	GGA	2
S	UCU	AGA	0
T	ACC	GGU	2
T	ACU	AGU	0
V	GUC	GAC	2
V	GUU	AAC	0
Y	UAC	GUA	3
Y	UAU	AUA	0

Table 9-7. Frequency distribution of tRNA genes for translating R-ending codons in *E. coli*. R-ending codons in each codon family are typically translated by two types of tRNAs, one with a wobble U and the other with a wobble C at the anticodon. Symbols as in Table 9-5.

AA	Codon	tRNA with anticodon	Freq
A	GCA	UGC	3
A	GCG	CGC	0
E	GAA	UUC	4
E	GAG	CUC	0
G	GGA	UCC	1
G	GGG	CCC	1
K	AAA	UUU	6
K	AAG	CUU	0
L	CUA	UAG	1
L	CUG	CAG	4
L	UUA	UAA	1
L	UUG	CAA	1
P	CCA	UGG	1
P	CCG	CGG	1
Q	CAA	UUG	2
Q	CAG	CUG	2
R	AGA	UCU	1
R	AGG	CCU	1
R	CGA	UCG	0
R	CGG	CCG	1
S	UCA	UGA	1
S	UCG	CGA	1
T	ACA	UGU	1
T	ACG	CGU	2
V	GUA	UAC	5
V	GUG	CAC	0

It is important to keep in mind that AT-rich genes may be transcribed more efficiently than GC-rich genes simply because the nucleotide pool typically features more A and T than G and C. Nucleotide C is particularly rare. ATP is much higher than that of the other three rNTPs (Colby and Edlin, 1970). For example, in the exponentially proliferating chick embryo fibroblasts in culture, the concentration of ATP, CTP, GTP and UTP, in the unit of (moles $\times 10^{-12}$ per 10^6 cells), is 1890, 53, 190, and 130, respectively, in 2-hour culture, and 2390, 73, 220, and 180, respectively, in 12-hour culture. Relative abundance of dNTP exhibits similar patterns. The surplus in A and deficiency in C has been proposed to affect RNA synthesis and DNA replication (Rocha and Danchin, 2002; Xia, 2005c; Xia *et al.*, 1996; Xia *et al.*, 2006; Xia and Yuen, 2005), although there is little direct evidence supporting the claim that the rate of RNA synthesis is in any way affected by AT% of the RNA.

If AT-rich genes indeed are transcribed more efficiently, then they will have high mRNA levels and at the same time low CAI values. The negative correlation between CAI and mRNA level has indeed been documented for AT-rich genes (dos Reis *et al.*, 2003; Jia and Li, 2005). This should highlight the point that CAI and CAAI are not measures of gene expression at the mRNA level because, at least for AT-rich genes, the indices are expected to be negatively correlated with the mRNA level (which unfortunately is often referred to as "gene expression" without the qualification of "at the transcription level").

Before we finish this section, the reader may wonder if, in this particular case, we may reasonably claim that "what is true for *E. coli* is also true for the human". We have shown the human tRNA data (Table 9-5), which would make sense of a negative correlation between CAI (or CAAI) and AT% in human genes. However, we have not yet seen such a negative correlation for human genes. So here I will just briefly mention that the negative correlation is present and statistically significant for not only human, but also the cow, the mouse and the budding yeast.

6. WILL AT-RICH MRNA BE TRANSLATED INEFFICIENTLY?

One highly pertinent question arising from the reasoning in the previous section is whether AT-rich mRNAs will be translated inefficiently because of the increased pairing of U-ending codons with the G at the tRNA anticodon wobble site. The answer to this question may not be obvious and requires some discussion from an evolutionary point of view.

According to the canonical codon-anticodon pairing rules first proposed for fungal mitochondria (Heckman *et al.*, 1980; Martin *et al.*, 1990), Y-ending codons can be wobble-translated by tRNA with a wobble G at the anticodon because G can not only pair with C but also wobble-pair with U, and R-ending codons by tRNA with a wobble U at the anticodon (through U/A and U/G pairing). To facilitate the exposition, let us designate the nucleotide at the anticodon wobble site as A^1, G^1, C^1 and U^1 respectively (where the superscript 1 indicates the wobble site at the first position of the tRNA anticodon), and that at the third codon position as A_3, G_3, C_3 and U_3, respectively. These canonical codon-anticodon pairing rules imply two Watson-Crick pairs, U^1/A_3 and G^1/C_3, as well as two wobble pairs, U^1/G_3 and G^1/U_3, respectively.

If U^1/G_3 pair is not as good as C^1/G_3 in term of translation efficiency and accuracy, then there should be selection in favor of the origin of tRNA genes with C^1 to eliminate the necessity of U^1/G_3 pairs in translation. Similarly, if G^1/U_3 is not as good as A^1/U_3, then selection should favor the origin of tRNA genes with A^1 to eliminate the necessity of G^1/U_3 pairs in translation.

Tables 9-5, 9-6 and 9-7 show that A-ending and G-ending codons are frequently translated by separate tRNAs with U^1 and C^1, respectively, but C-ending and U-ending codons are almost always translated by tRNA with only G^1. This implies that G^1/U_3 pairs may be as good as perfect Watson-Crick pairs (e.g., A^1/U_3, and G^1/C_3), i.e., selection for the origin of tRNA genes with A^1 to reduce or eliminate G^1/U_3 pairs is weak. In contrast, wobble U^1/G_3 likely is not as good as perfect Watson-Crick pairing, and selection has resulted in the origin of tRNAs with C^1 to replace the U^1/G_3 pair by the C^1/G_3 pair in translation. In other words, both C^1 tRNA and U^1 tRNA are needed to translate G-ending and A-ending codons, respectively.

If the reasoning above is correct, i.e., if G^1/U_3 pairs are as good as Watson-Crick pairs, then an increase in U-ending codons with a consequent increase in G^1/U_3 pairs during translation of AT-rich genes does not necessarily imply inefficient translation, although such an increase would still reduce CAI or CAAI. If this is true, then CAI and CAAI are not good measures of translation efficiency for AT-rich genes.

On the other hand, if G^1/U_3 pairs are not as good as Watson-Crick pairs but A^1 tRNA did not originate because of some unknown evolutionary constraints, then AT-rich genes may be translated inefficiently and the negative correlation between CAI (and CAAI) and AT% does not invalidate their application to AT-rich genes. An extension of this argument is that if AT-rich genes cannot be translated efficiently, yet the cell requires a large quantity of proteins from some of these genes, then the only way to meet the cellular need is to increase transcriptional efficiency to produce many mRNAs. This would also contribute to the association of low CAI (or CAAI) and high gene expression at the transcription level. As I have mentioned before, such negative correlation between the two has already been documented (dos Reis *et al.*, 2003; Jia and Li, 2005). This is another warning against interpreting CAI (or CAAI) as an index of gene expression at the transcriptional level.

7. CODON ADAPTATION INDEX AND PROTEOMICS: CLARRIFICATION OF SOME MISUNDERSTANDINGS

Proteins are involved in all three essential biological processes, i.e., DNA replication, transcription and translation. Their functions include metabolic interactions, signal transduction, gene regulation, transport, cellular structures, organelle constituents, storage reserves, protection, and cellular homeostasis. Normal function of living cells depends much on normal production of proteins, and many of the known human genetic diseases are caused by the overproduction or underproduction of certain proteins. Many human genetic diseases can be attributed to the overproduction or underproduction of certain proteins. Therefore, proteins constitute one of the most important cellular components in the cell, and the understanding of their interactions is believed to be the key to many unresolved biological problems today, including cancer (Chen *et al.*, 2005; Sparre *et al.*, 2005).

Given the importance of proteins in understanding cellular functions, a biologist naturally would wish to characterize the protein production, especially the temporal change of protein production and their interactions (Figeys, 2002; Sloane *et al.*, 2002; Wilson and Nock, 2002), in the living cells. However, it has been difficult to characterize the expressed proteome in spite of the recent advances in protein separation, identification and quantification (Figeys, 2003b, 2003a; Vasilescu and Figeys, 2006; Washburn *et al.*, 2001; Yates, 2004a, 2004b). Large-scale proteomics is handicapped by the difficulties in (1) separating certain proteins, especially hydrophobic membrane proteins and those with extreme isoelectric points and (2) identifying rare proteins (Chen *et al.*, 2005; Gygi *et al.*, 1999; Kolkman *et al.*, 2005; Sparre *et al.*, 2005; Tian *et al.*, 2004).

One alternative is to characterize the expressed transcriptome and predict the proteome from the transcriptome. The ease in generating transcriptomic data by the microarray (Diehn *et al.*, 2000; Epstein and Butow, 2000; Gaasterland and Bekiranov, 2000; Holstege *et al.*, 1998; Schena, 1996; Schena, 2003) or SAGE (Madden *et al.*, 1997; Saha *et al.*, 2002; Velculescu *et al.*, 1995; Velculescu *et al.*, 1997; Zhang *et al.*, 1997) experiments has fostered the hope that protein production can be predicted from transcriptomic data. In particular, the reproducibility of the SAGE method has been shown to be excellent (Dinel *et al.*, 2005).

However, the relationship between the mRNA and protein levels in cells is either poor or moderate (Baliga *et al.*, 2002; Chen *et al.*, 2002; Futcher *et al.*, 1999; Griffin *et al.*, 2002; Gygi *et al.*, 1999; Ideker *et al.*, 2001; Tian *et al.*, 2004). The best result was obtained by using the transcriptomic data obtained by SAGE (Velculescu *et al.*, 1997) and the proteomic data acquired by 2D-gel electrophoresis and capillary liquid chromatography-tandem mass spectrometry (Gygi *et al.*, 1999), with a correlation of 0.93 between the mRNA abundance and the protein abundance. However, this reported correlation of 0.93 is misleading because it results mainly from a few outlying points with very high mRNA abundance and protein abundance. This problem is illustrated in Figure 9-7.

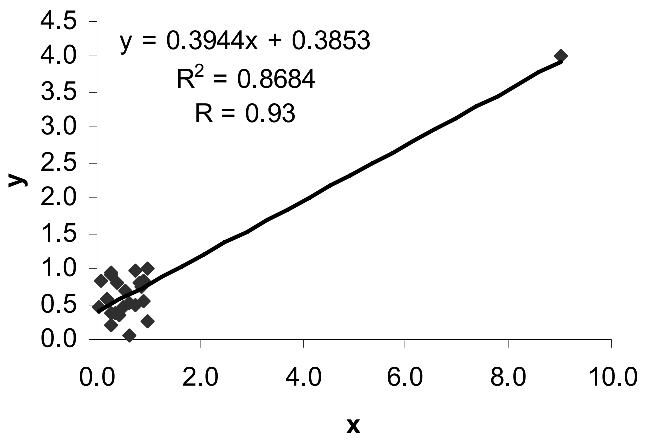


Figure 9-7. Inappropriate use of Pearson correlation in case of outliers. The figure has 23 points with 22 being randomly generated. The correlation is near 0 when the outlying point is removed.

It is statistically inappropriate to use such a correlation coefficient to characterize the relationship between two variables (e.g., between the mRNA abundance and the protein abundance) with outliers. The correlation coefficient between mRNA abundance and protein abundance drops quickly from 0.93 to around 0.3 when the few extremely highly expressed genes were removed (Gygi *et al.*, 1999).

While transcriptomic data measure the number of mRNA molecules in the cell, it is not expected to be a good predictor of protein production by itself. Two protein-coding genes may have the same mRNA level, but one may be translated more efficiently than the other and consequently produce more proteins. Translation efficiency is partially reflected by the codon usage and amino acid usage. If a mRNA uses codons recognized by most abundant tRNA, and if the resulting protein is made mostly of abundant (typically energetically cheap) amino acids, then we expect more proteins produced from this mRNA than an alternative mRNA whose codons are recognized by few tRNA and whose resulting proteins are made mostly of rare and energetically expensive amino acids. For this reason, codon usage bias and amino acid usage can improve significantly the prediction of protein abundance, especially in predicting protein abundance with low mRNA abundance. CAI measures the component of translation efficiency reflected by codon usage bias and is expected to contribute to predicting protein production.

There are often misunderstandings of CAI. For example, it was thought "that codon bias is a measure of protein abundance" (Gygi *et al.*, 1999) and expected to be able to predict protein production by itself. The codon usage bias in Gygi et al. (1999) is CAI (Sharp and Li, 1987) taken from the yeast proteome database (YPD) (Hodges *et al.*, 1999). As we know now, CAI is not a measure of protein abundance. Instead, it is a measure of how well a coding sequence is adapted to the translation machinery (Bulmer, 1991; Ikemura, 1981; Sharp and Li, 1987; Xia, 1998b). If two protein-coding genes of equal lengths have the same copy number of their mRNA, then the gene with a higher CAI value is expected to be translated more efficiently than the one with a lower CAI, everything else being equal. CAI does not indicate whether a gene will be transcribed or not. A gene may be highly expressed in one particular tissue at one particular time, but it may be turned off in other tissues or at other times, and consequently has no mRNA in the cell. Thus, the codon usage bias should not be used alone to predict protein abundance as previously suggested (Futcher *et al.*, 1999), but instead should be incorporated into a model together with the mRNA abundance to predict protein abundance.

The misunderstanding of CAI by Gygi et al (1999) led to wrong data analysis and wrong conclusions. I particularly wish to highlight the significance of the result concerning gene ENO1 because the mRNA and protein abundance of this gene was used by Gygi et al. (1999) to support two major claims in their paper. First, they concluded that mRNA abundance is insufficient to predict the protein abundance. This claim is also echoed by others (Griffin *et al.*, 2002; Tian *et al.*, 2004). Instead of providing statistical substantiation, Gygi et al. (1999) provided a concrete example to show that genes with similar mRNA abundance such as ENO1 and FRS2 both with the mRNA abundance value of 0.7 differ in protein abundance by more than 20-fold (e.g., the protein abundance values for ENO1 and FRS2 are 44.2 and 2.3, respectively).

Although ENO1 and FRS2 do not differ in mRNA abundance, they differ greatly in CAI which is 0.93 and 0.451 for ENO1 and FRS2, respectively. In other words, ENO1 mRNA is expected to be translated into proteins much more efficiently than FRS2. Given the same mRNA abundance of 0.7, we naturally should expect the ENO1 mRNA to be translated into many more copies of proteins than the FRS2 mRNA. This is exactly what has been observed, a vindication of incorporating codon usage bias in a model for predicting the protein abundance. In fact, there are 13 genes with the lowest average mRNA abundance of 0.7 (Gygi *et al.*, 1999), and ENO1 is the only gene with a very high CAI value and also a high protein abundance value of

44.2. The predicted protein abundance value for ENO1 is almost exactly the same as the observed value when both mRNA and CAI are incorporated into the prediction (Xia, 2005b). So there is nothing extraordinary for ENO1 to have its protein abundance much higher than other genes with similar mRNA abundance. One should not use its high protein production relative to FRS2 as evidence to support the claim that mRNA abundance is a poor predictor of protein abundance.

The second major claim made by Gygi et al. (1999) is that codon usage bias is a poor predictor of protein production because ENO1 and ENO2 both have high CAI values but the protein abundance value for the latter is much greater than the former. As mentioned before, codon usage bias says nothing about whether a gene will be transcribed or not. It measures how efficient a transcribed mRNA can be translated by the translation machinery in the cell. As Gygi et al. (1999) has recognized, ENO1 is down-regulated, and ENO2 up-regulated in transcription, in their culture conditions, with the former has few transcripts relative to the latter (0.7 and 289, respectively). The observation that they have different protein abundance does not in any way belittle the utility of codon usage bias in predicting protein abundance. In fact, the predicted protein abundance value for ENO2 is also quite similar to its observed value (Xia, 2005b), i.e., consistent with the model that both mRNA abundance and codon usage bias are important predictors of protein abundance. As the previous contrast between ENO1 and FRS2 has shown, codon usage bias is crucial in supplying the answer missed by mRNA abundance alone.

This illustration above vindicates the wisdom of R. A. Fisher that "No aphorism is more frequently repeated in connection with field trials, than that we must ask Nature few questions, or ideally, one question at a time. The writer is convinced that this view is wholly mistaken. Nature, he suggests, will respond to a logical and carefully thought-out questionnaire; indeed, if we ask her a single question, she will often refuse to answer until some other topic has been discussed" (Fisher, 1926). When Gygi et al. (1999) asked if CAI could predict protein production, nature refused to give them the right answer. When they asked if mRNA abundance could predict protein production, they again were given a wrong answer. However, when both mRNA and CAI were taken into account to predict protein production, a very good answer was given (Xia, 2005b).

The example above reminded me of an anecdote involving the former Chinese premier Zhou Enlai that highlights the importance of having a global view of things. After the establishment of the People's Republic of China, the government launched a nationwide campaign against pornography and prostitution. The effort resulted in a complete eradication of prostitution in mainland China. In a news conference celebrating this major socialist achievement, a western reporter suddenly asked Premier Zhou if China still had prostitutes. All Chinese officials present were surprised to hear Premier Zhou answering “Yes”, and all applauded when Premier Zhou added that “They are in Taiwan”. It turned out that the western reporter was hoping that Premier Zhou would answer “No” given the special occasion so that he could then claim that Chinese leadership did not consider Taiwan as part of China. A global view of things by Premier Zhou saved the day. It is time for biologists to take a global view of the cell instead of asking nature a single question at a time.

As I mentioned before, the utility of an index of translation efficiency ultimately depends on its ability to predict protein abundance. Two previous publications (Futcher *et al.*, 1999; Gygi *et al.*, 1999) have evaluated the utility of CAI by studying its relationship to protein production. Futcher et al. (1999) noted that log-transformation of the protein abundance data linearized its relationship with CAI and stabilized variance of protein abundance over the range of CAI. I present in Table 9-8 Pearson correlation coefficients between the log-transformed data and the two indices of codon usage bias, CAAI and CAI. Two points are worth noting. First, there is a clear relationship between the two variables, indicating that these indices of codon usage bias can contribute to a better prediction of protein abundance. Second, CAAI consistently exhibits a highly correlation than the conventional CAI.

Table 9-8. Pearson correlation coefficients between protein abundance and indices of codon usage CAAI and CAI.

	lnGygi(2)	lnFut1(2)	lnFut2(2)
CAAI	0.7072	0.8464	0.7372
CAI-Gygi(1)	0.6989	0.8055	0.7229
CAI-Futcher(1)	0.7024	0.7736	0.5935

(1) CAI values differ slightly between Futcher et al. (1999) and Gygi et al. (1999), likely caused by using slightly different reference sets (which were not specified in the papers).

(2) Log-transformed protein abundance in Futcher et al. (1999) and Gygi et al. (1999). Protein abundance was measured by Futcher et al. (1999) in glucose and ethanol media, designated lnFut1 and lnFut2, respectively in the Table.

The misunderstanding of codon usage bias by Gygi et al. (1999) and their conclusion that codon usage bias is a poor predictor of protein abundance have the unfortunate consequence that several subsequent studies of the relationship between mRNA and protein levels have ignored codon usage bias all together (Baliga *et al.*, 2002; Chen *et al.*, 2002; Griffin *et al.*, 2002; Ideker *et al.*, 2001; Tian *et al.*, 2004) by citing conclusions in Gygi et al. (1999). It seems that science can move backwards quite easily.

Chapter 10

PROTEIN ISOELECTRIC POINT

1. INTRODUCTION

Proteins have ionizable groups such as carboxyl groups and amino groups. Since the charge of these groups depends on pH, a protein molecule can have different charges at different pH. The isoelectric point of a protein is the pH at which the protein carries not net charge.

Why should a book entitled "bioinformatics and cell" have a chapter on protein isoelectric point (pI) and its computation? There are several answers to this question, all related to the aphorism that proteins are the workhorses in the cell and that an understanding of a living cell cannot come into shape without a good understanding of proteins housed in the cell.

First, pI is important in understanding enzyme-substrate interactions. An enzyme and its substrate should not be both positively charged or both negatively charged because the two will repulse each other. To know whether the enzyme is positively charged or negatively charged at a given ambient pH, we need to know pI of the protein. The protein is positively charged if its pI is greater than the pH and negatively charged if its pI is smaller than the pH.

Second, biologists have tried for a long time to characterize and analyze proteins in living cells. Large-scale proteomic research started with sodium dodecyl sulfate polyacrylamide gel electrophoresis or SDS-PAGE (Laemmli, 1970). Subsequent perfection of isoelectric focusing leads to the development of 2D-SDS-PAGE. While large-scale peptide analysis methods have been developed by John Yates and colleagues recently (Washburn *et* *al.*, 2001; Yates, 2004a, 2004b), 2D-SDS-PAGE remains the most frequently used proteomic method. Indeed, 2D-SDS-PAGE has almost become synonymous to proteomic research (Liebler *et al.*, 2002, p. 36). Understanding 2D-SDS-PAGE requires an understanding of pI. For example, if the pI values of proteins in a cell ranges from 2 to 14 and you intend to use an isoelectric-focusing strip with a fixed pH range between 3 and 7 in your 2D-SDS-PAGE, then you know that you will miss many proteins. It is now routine for researchers to extract all annotated coding sequences in a genome, translate them into proteins, obtain their molecular mass and theoretical pI values, and generate an *in silico* 2D-SDS-PAGE to improve the experimental design of a real 2D-SDS-PAGE. This *in silico* 2D-SDS-PAGE can also be used as the expected pattern to compare with the observed pattern on a real 2D-SDS-PAGE. Those proteins found at the same location in both the *in silico* gel and the real gel may be assumed to have undergone no posttranslational modification, whereas those whose coordinates do not match between the *in silico* gel and the real gel are good candidates for studying posttranslational modification.

Third, the stability of a protein often depends on the electrostatic interaction between its positively charged and negatively charged groups on the surface of protein at its physiological condition. When the pH deviates substantially from the physiological pH, the electrostatic interaction is disrupted and the protein will denature. For example, at extremely low pH (acidic), the carboxyl group is protonated and negative charges are decreased, whereas more of the amino groups are positively charged. Such a protein will tend to lose its stability, become less compact and finally denature.

Fourth, if a highly expressed protein happens to have its pI equal to the cytoplasmic pH, then there is no electrostatic repulsion among different copies of this protein when it is mass produced. Because the protein is not charged, its solubility is the lowest, and different copies of this protein may aggregate and precipitate, which is often bad for the cell. The "amyloid precursor protein" causing Alzheimer disease and the prion protein causing the mad cow disease are examples of the undesirable protein aggregation and precipitation. From an evolutionary point of view, one should expect directional selection driving the protein pI away from the physiological pH, and the directional selection should be strong in highly expressed proteins than in lowly expressed proteins. There are known cases where natural selection has shaped protein pI. For example, *Helicobacter pylori,* a bacterial species colonizing mammalian stomach, features a set of membrane proteins that are positively charged, and this positively charged membrane is likely important in alleviating the influx of protons (H^+) in the acidic stomach fluid into the bacterial cytoplasm (Sachs *et al.*, 2003; Xia and Palidwor, 2005).

This chapter begins with a brief review of the basic concepts of biochemistry related to computing protein pI. The method of computing protein pI based on computer iterations is then presented. This is then followed by special sections illustrating bioinformatics applications of pI.

2. AMINO ACID AND PROTEIN ISOELECTRIC POINT

We need to know the ionization constant (K_a) and a few associated concepts. K_a measures the tendency of a chemical to give up proton. With an ionizing reaction below involving a weak acid:

$$\mathrm{RCOOH{=}RCOO^- + H^+} \tag{10.1}$$

K_a is expressed as

$$K_a = \frac{[\mathrm{RCOO^-}][\mathrm{H^+}]}{[\mathrm{RCOOH}]} \tag{10.2}$$

which will be re-designated as K_{a1} hereafter for the ionization constant in an amino acid ionization reaction involving a weak acid. The ionization constant involving a weak base will be designated as K_{a2}.

Take the base-10 logarithm (lg) of both sides of Eq. (10.2), we have

$$\lg K_{a1} = \lg[\mathrm{H^+}] + \lg\frac{[\mathrm{RCOO^-}]}{[\mathrm{RCOOH}]} \tag{10.3}$$

Recall that, in chemistry, pH is defined as $-\lg[\mathrm{H^+}]$ and pK_{a1} as $-\lg K_{a1}$. So the equation above becomes the well known Henderson-Hasselbalch equation:

$$pH = pK_{a1} + \lg\frac{[\mathrm{RCOO^-}]}{[\mathrm{RCOOH}]} \tag{10.4}$$

For weak bases such as amines, the Henderson-Hasselbalch equation is

$$pH = pK_{a2} + \lg\frac{[\mathrm{NH_2}]}{[\mathrm{NH_3^+}]} \tag{10.5}$$

An amino acid is a weak base and a weak acid at the same time. It exists as a cation at low pH, and an anion at high pH. The amino acid at the intermediate pH when it carries no net charge (i.e., the positive and the negative charge cancel each other) is called a zwitterion:

$$NH_3^+\text{-RCH-COOH} \overset{pK_{a1}}{\rightleftarrows} NH_3^+\text{-RCH-COO}^- \overset{pK_{a2}}{\rightleftarrows} NH_2\text{-RCH-COO}^-$$

The isoelectric point of an amino acid is defined as the pH at which the amino acid exists as a zwitterion. Thus, given the condition that [RCOO-] in Eq. (10.4) equals [NH_3^+] in Eq. (10.5), i.e., the negative charge equals the positive charge, we have

$$\begin{aligned}
&\text{pH} - \text{p}K_{\text{a}_1} + \lg[\text{RCOOH}] = \text{p}K_{\text{a}_2} - pH + \lg[\text{NH}_2] \\
&2\text{pH} = \text{p}K_{\text{a}_1} + \text{p}K_{\text{a}_2} + \lg[\text{NH}_2] - \lg[\text{RCOOH}] \\
&\quad = \text{p}K_{\text{a}_1} + \text{p}K_{\text{a}_2} + \lg([\text{AA}] - [\text{NH}_3^+]) - \lg([\text{AA}] - [\text{RCOO}^-]) \qquad (10.6) \\
&\quad = \text{p}K_{\text{a}_1} + \text{p}K_{\text{a}_2} \\
&\text{pH} = \frac{\text{p}K_{\text{a}_1} + \text{p}K_{\text{a}_2}}{2}
\end{aligned}$$

where [AA] is the total concentration of the amino acid. The final pH in Eq. (10.6) is defined as the isoelectric point of the amino acid and is designated by a new symbol pI, i.e., it is the pH at which [RCOO-] = [NH_3^+]. The pK_a values for the ionizable residues are listed in Table 10-1.

Table 10-1. The ionizable residues in proteins and their approximate pK_a values. The last four columns illustrate the calculation of pI for protein polA2 from *Halobacteria* NRC1.

Amino acid group	$pK_a^{(1)}$	$pK_a^{(2)}$	N	pH = 3	pH = 4.2227	pH = 5
Arginine	12.5	12.50	95	95.0000	95.0000	95.0000
Lysine	10.0	10.79	31	31.0000	31.0000	30.9999
Histidine	6.0	6.50	38	37.9880	37.8004	36.8352
Tyrosine	10.4	10.95	40	0.0000	0.0000	0.0000
Cysteine	8.3	8.30	18	-0.0001	-0.0015	-0.0090
Glutamic acid	4.1	4.25	115	-6.1226	-55.6905	-97.6374
Aspartic acid	4.1	3.91	161	-17.6374	-108.2869	-148.8972
N-terminal α-amino	8.0	8.56	1	1.0000	1.0000	0.9997
C-terminal α-carboxyl	3.1	3.56	1	-0.2159	-0.8214	-0.9650

(1) From Berg et al. (2002).

(2) The average of 16 sets of values that I collected from journal papers, books and web pages

Each peptide has a H_2N- group at one end and a –COOH group at the other end, and its charge depends mainly on the ionization of the side chain of amino acid residues (Table 10-1). Note that the pK_a values depend on temperature, ionic strength and other less well defined factors, so that values in Table 10-1 are approximate. Also note that there are only seven amino acids that have ionizable residues. Other amino acid residues are irrelevant in computing protein pI.

The pI of a protein is typically computed by an iterative method illustrated in Table 10-1, with the polA2 protein from *Halobacteria* NRC1 and the pK_a values in the third column in Table 10-1. First, one counts the frequencies of the seven ionizable amino acid residues in Table 10-1 (shown in the column headed with 'N'). Next, for each amino acid, we compute the proportion of positively charged and negatively charged residues, designated by P_{NH3+} and P_{RCCO-}, respectively:

$$\begin{aligned} P_{RCOO^-} &= \frac{[RCOO^-]}{[RCOO^-]+[RCOOH]} \\ P_{NH_3^+} &= \frac{[NH_3^+]}{[NH_3^+]+[NH_2]} \end{aligned} \qquad (10.7)$$

From equations (10.4) and (10.5), we can derive these two proportions as follows:

$$\begin{aligned} &\lg\frac{[\mathrm{RCOO^-}]}{[\mathrm{RCOOH}]} = pH - pK_\mathrm{a} \\ &\frac{[\mathrm{RCOO^-}]}{[\mathrm{RCOOH}]} = 10^{pH-pK_\mathrm{a}} \\ &P_{\mathrm{RCOO^-}} = \frac{10^{pH-pK_\mathrm{a}}}{1+10^{pH-pK_\mathrm{a}}} \end{aligned} \qquad (10.8)$$

$$\begin{aligned} &\lg\frac{[\mathrm{NH_2}]}{[\mathrm{NH_3^+}]} = pH - pK_\mathrm{a} \\ &\frac{[\mathrm{NH_2}]}{[\mathrm{NH_3^+}]} = 10^{pH-pK_\mathrm{a}} \\ &P_{\mathrm{NH_3^+}} = \frac{1}{10^{pH-pK_\mathrm{a}}+1} \end{aligned} \qquad (10.9)$$

P_{RCOO-} in Eq. (10.8) is interpreted in two similar ways. For example, with N Glu residues, P_{RCOO-} means the proportion of the residues that carry the negative charge at a given pH. For a single Glu residue, P_{RCOO-} means the probability that the residue will be in a negatively charged state. Similarly, for N Arg residues, $P_{NH3}+$ means the proportion of the residues that are positively charged at a given pH. For a single Arg residue, $P_{NH3}+$ means the probability that the residue will be in a positively charged state.

With the 95 Arg residues in Table 10-1, the number of positively charged Arg residues, given pH = 3 is

$$N_{Arg^+} = N_{Arg} P_{\mathrm{NH}_3^+} = 95 \times \frac{1}{10^{3-12.50}+1} = 95 \tag{10.10}$$

Similarly, the number of negatively charged Asp residues out of the total of 161 in Table 10-1, given pH = 3, is

$$N_{Asp^-} = N_{Asp} P_{\mathrm{RCOO}^-} = 161 \times \frac{10^{3-3.91}}{1+10^{3-3.91}} = 17.63745 \tag{10.11}$$

Such calculations are done for each of the amino acids to generate the forth column in Table 10-1 headed by ‘pH = 3’. The negative sign is used to indicate those that carry negative charges. The summation of the column is 141.0119, which means that the protein is positively charged at pH = 3.The iterative procedure then finds a pH value at which the protein is negatively charged. Suppose the next value is pH = 5 and we repeat the procedure to generate the last column in Table 10-1 headed by ‘pH = 5’. Now the summation becomes -83.6737, i.e., the protein is negatively charged at pH = 5. Now we know that the pH at which the protein carries no net charge must lie between 3 and 5. Suppose we happen to be lucky to try pH = 4.222657 and repeat the procedure to generate the fifth column in Table 10.1. Now the sum of the values is very close to 0, i.e., the positively charged and negatively charged residues cancel each other. Therefore, pI = 4.222657. Many efficient numerical algorithms are available to find pI (Press *et al.*, 1992) to any degree of accuracy. The software DAMBE (Xia, 2001; Xia and Xie, 2001b) implements this iterative procedure to compute the protein pI.

3. GENOMIC PROFILING OF PROTEIN ISOELECTRIC POINT: A CASE STUDY WITH *HELICOBACTER PYLORI*

Genomic pI profiling refers to the computation and graphic display of pI for all genome-derived proteins. Figure 10-1 is a genomic pI profiling for *Escherichia coli.* The saddle-shaped distribution is typical of most species from prokaryotes to eukaryotes. There are several reasons for such a saddle-shaped distribution, but we will postpone the presentation of the reasons to the last section of the chapter.

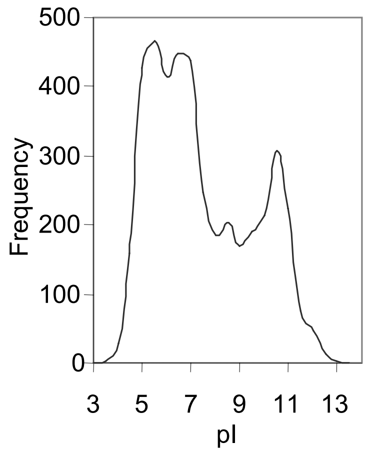


Figure 10-1. Frequency distribution of pI values of genome-derived proteins from *E. coli.*

The genomic pI profile for the bacterial pathogen, *Helicobacter pylori* (Figure 10-2) is quite different from that of *E. coli.* The conspicuous peak of basic proteins has been interpreted as a mechanism protecting the organism against its acidic environment, i.e., the mammalian stomach (Sachs *et al.*, 2003; Xia and Palidwor, 2005). Note that a protein is acidic if its pI is smaller than 7 and basic if its pI is greater than 7. However, whether the protein is positively or negatively charged depends on its environmental pH. A protein will be positively charged when the environmental pH is lower than its pI, and negatively charged when the environmental pH is higher than its pI. For *H. pylori,* its environmental pH is near 1 but its cytoplasmic pH can be maintained around pH near 5. Most of its proteins have their pI greater than its cytoplasmic pH and are consequently positively charged.

Why does the pI profile of *H. pylori* miss the conspicuous peak visible in the pI profile of *E. coli* around pH = 5? From an evolutionary point of view, when we see differences among different species, we typically think in two ways. The first is that the pI profile has little to do with the survival and reproduction of the species and can drift in any way and take up any shape. The alternative hypothesis is that the differences between the species are respectively beneficial to the organisms living in different environments. It is in this context that we will examine in detail the pI profile of *H. pylori.*

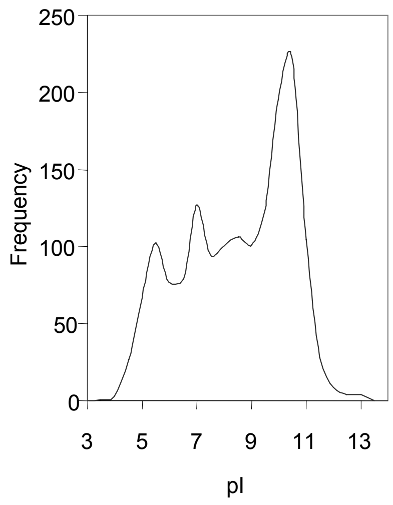


Figure 10-2. Frequency distribution of pI values of genome-derived proteins from *H. pylori.*

H. pylori (Figure 10-3) is one of the terminal lineages in the highly invasive *Helicobacter* complex. It thrives in the acidic environment of mammalian stomach, causing gastric and duodenal ulcers and gastric cancer in human (Correa, 1997; Hamajima *et al.*, 2004; Hunt, 2004; Menaker *et al.*, 2004; Siavoshi *et al.*, 2004).

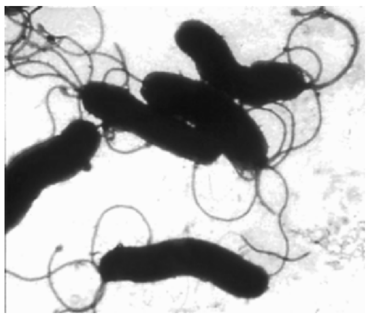


Figure 10-3. Microscopic image of *H. pylori* (From Paul Stokes Hoffman, University of Virginia).

Being an acid-resistant neutralophile (Bauerfeind *et al.*, 1997; Rektorschek *et al.*, 2000; Sachs *et al.*, 1996; Scott *et al.*, 2002), *H. pylori* is capable of surviving for at least 3 hours at pH 1 with urea (Stingl *et al.*, 2001) and maintaining a nearly neutral cytoplasmic pH between pH 3.0 and 7.0 (Matin *et al.*, 1996; Scott *et al.*, 2002; Stingl *et al.*, 2002b). These properties allow it to survive and reproduce in the human stomach where the gastric fluid has a pH averaging about 1.4 over a 24-h period (Sachs *et al.*, 2003).

The buffering action of the gastric epithelium and limited acid diffusion through the gastric mucus were previously thought to protect the bacterium against stomach acidity, but both empirical studies (Allen *et al.*, 1993) and theoretical modeling (Engel *et al.*, 1984) have suggested that the protection is rather limited (Matin *et al.*, 1996; Sachs *et al.*, 2003). Recently it has also been shown that mucus does not hinder proton diffusion and a trans-mucus pH gradient is abolished when the luminal pH drops to < 2.5 (Baumgartner and Montrose, 2004). It is therefore necessary for *H. pylori* to have acid-resistance mechanisms to colonize the gastric mucosa successfully (Sachs *et al.*, 2003).

H. pylori has evolved two mechanisms protecting itself against the acidic environment in the mammalian stomach. The first involves the urease gene cluster *ureABIEFGH*. The constitutively expressed cytoplasmic urease, coded by *ureAB*, catalyzes urea to generate $2NH_3 + CO_2$ to buffer against the H^+ influx into either the periplasm or the cytoplasm (Mobley *et al.*, 1991; Rektorschek *et al.*, 2000; Sachs *et al.*, 2003; Stingl *et al.*, 2002a) and to facilitate the extrusion of H^+ from the cytoplasm in the form of NH_4^+ (Stingl *et al.*, 2002a). However, urease is an apoenzyme requiring a nickel to be active. The *ureEFGH* gene cluster, whose expression is acid-induced, codes for nickel-sequestrating proteins that insert nickel into the urease, leading to increased and sustained urease activity (Sachs *et al.*, 2003; Wen *et al.*, 2003; Williams *et al.*, 1996).

The urease, once activated, naturally needs a constant supply of urea as its substrate, and the cell has two sources of urea supply, one intrinsic and one extrinsic. The extrinsic source refers to urea present in saliva and stomach fluid. The exposure to gastric acid results in a large increase in urea influx into the cell due to the pH-gating of the urea channel protein UreI (Bury-Mone *et al.*, 2001; Weeks *et al.*, 2000). The intrinsic source comes from efficient conversion of arginine to urea in the cytoplasm by the highly expressed arginase in *H. pylori* (Mendz and Hazell, 1996). For this reason, arginine is underused in *H. pylori* proteins and the positively charged membrane in *H. pylori* is mainly maintained by a surplus of positively charged lysine residues (Xia and Palidwor, 2005).

The second acid-resistant mechanism in *H. pylori* is the restriction of acute proton entry across its membranes by having a high frequency of positively charged amino acids in the inner and outer membrane proteins (Sachs *et al.*, 2003; Scott *et al.*, 1998; Valenzuela *et al.*, 2003). This is supported by recent discovery of a basic proteome (Tomb *et al.*, 1997), a set of basic membrane proteins (Baik *et al.*, 2004) in *H. pylori*, and an extensive genomic analysis (Xia and Palidwor, 2005).

The membrane proteins have long been suspected to play an important role in acid resistance in *H. pylori* (Alm *et al.*, 2000; Huynen *et al.*, 1998; Solnick *et al.*, 2004; Yamaoka *et al.*, 2002). In a recent characterization of 34 membrane proteins of *H. pylori* STR 26695 (Baik *et al.*, 2004), four proteins (HP0243, HP0072, HP0512 and HP1563) have pI values ranging from 5.86 to 6.25, whereas the rest 30 have pI greater than 7. The average pI is 8.9221 for these 34 membrane proteins, whereas the average calculated pI value for the other 1542 proteins annotated in the genomic sequence (NC_000915) is 8.2147. The two average pI values are significantly different by a two-sample t-test (DF = 1572, t = 2.075, p = 0.0382, two-tailed test). Thus, membrane proteins are significantly more basic than the rest of the proteins in *H. pylori*.

A comparison of the *H. pylori* membrane proteins with those in the related *H. hepaticus,* may shed light on whether the basic membrane proteins in *H. pylori* result from adaptation in response to the acidic environment. If *H. hepaticus,* which is not acid resistant, also features a set of equally basic or even more basic membrane proteins, then the set of basic membrane proteins in *H. pylori* is likely an ancestral trait evolved before *H. pylori* became a stomach parasite and consequently should not be interpreted as resulting from adaptation in response to the acidic stomach environment. In contrast, if the set of basic membrane proteins is unique in *H. pylori,* then we would have more confidence in interpreting the basic membrane proteins as an adaptation.

BLASTing (Altschul *et al.*, 1990; Altschul *et al.*, 1997) those 34 *H. pylori* membrane proteins by their corresponding CDSs against the 1875 CDSs in the related *H. hepaticus* genome, with a cutoff e-value of 0.0001, revealed several interesting patterns (Xia and Palidwor, 2005). The four *H. pylori* membrane proteins with pI < 7 all have homologs in the *H. hepaticus* genome (NC_004917). In contrast, among the rest of 30 membrane proteins with pI > 7, only one has a homolog in the *H. hepaticus* genome. It is important to note that, out of the 1576 predicted protein-coding genes in the genomes of *H. pylori* strain 26695, 938 found matches in the genome of *H. hepaticus.* Similarly, 941 out of 1492 predicted protein-coding genes in the genome of *H. pylori* strain J99 (NC_000921) have matches in the genome of *H. hepaticus* (Suerbaum *et al.*, 2003). With this reference in mind, the number of matches for *H. hepaticus* 26695 membrane proteins in the *H. hepaticus* genome is relatively small. This suggests that nearly all those positively charged membrane proteins in *H. pylori* are unique, and most likely result from the evolution along the *H. pylori* lineage. These basic membrane proteins may either have evolved quickly along the *H. pylori* lineage so as to be beyond recognition in the *H. hepaticus* genome, or represent differential gain of genes in the *H. pylori* lineage or differential loss of genes in the *H. hepaticus* lineage. In any case, this result lends more support to the interpretation that the basic membrane proteins and the functional consequence that they alleviate the influx of H^+ into the *H. pylori* cell have resulted from adaptation to the acidic environment.

The interpretation above appears rather straightforward. However, science is never so simple. Almost any observed pattern in science can have multiple interpretations that seem perfectly consistent with the data. While two other alternative hypotheses, i.e., preadaptation and exaptation, have been evaluated against the adaptation hypothesis for the origin and maintenance of a basic proteome in *H. pylori* before (Xia and Palidwor, 2005), there is at least another hypothesis that has not been examined.

4. AN ALTERNATIVE EXPLANATION OF *H. PYLORI* DATA

As mentioned previously, a protein in a solution with a pH equal to the protein pI is not charged. If highly expressed proteins happen to have their pI equal to the cytoplasmic pH, then there is no electrostatic repulsion among these proteins when they are mass-produced. Because the proteins are not charged, their solubility is at the lowest, and they may aggregate and precipitate, which is often harmful to the cell. The "amyloid precursor protein" causing Alzheimer disease and the prion protein causing the mad cow disease are examples of the undesirable protein aggregation and precipitation.

A prediction arising from this simple observation is that organisms tend to avoid having proteins, especially those highly expressed ones, with their pI equal to intracellular pH because of the negative (purifying) selection against protein precipitation. Because most living organisms have physiological pH nearly neutral, their pI profiles should exhibit a saddle-shaped curve with relatively few proteins at their physiological pH but with a peak at the acidic pH range and another peak at the basic pH range. For *E. coli* living in mammalian intestine where the pH is about 8-9, we should expect relatively few proteins with pI in the range of 8-9, which is true (Figure 10-1).

One may note that the "valley" of the pI profile for *E. coli* (Figure 10-1) is not all that shallow, i.e., there are still many proteins that have theoretical pI in the range of 8-9. Will these proteins precipitate upon translation? There are two possible explanations. One is that some proteins will immediately be modified to assume a different pI, and the other is that these proteins are not mass produced and the chance of them forming aggregations and precipitate is consequently small. We can have a quick examination of the latter by plotting CAI or CAAI against pI, with the expectation that proteins with pI values in the range of 8-9 should not have high CAI values. Empirical data appear to support this expectation (Figure 10-4). Among proteins with CAI values greater than 7 and presumably are highly expressed, few have pI values in the range of 8-9 (Figure 10-4).

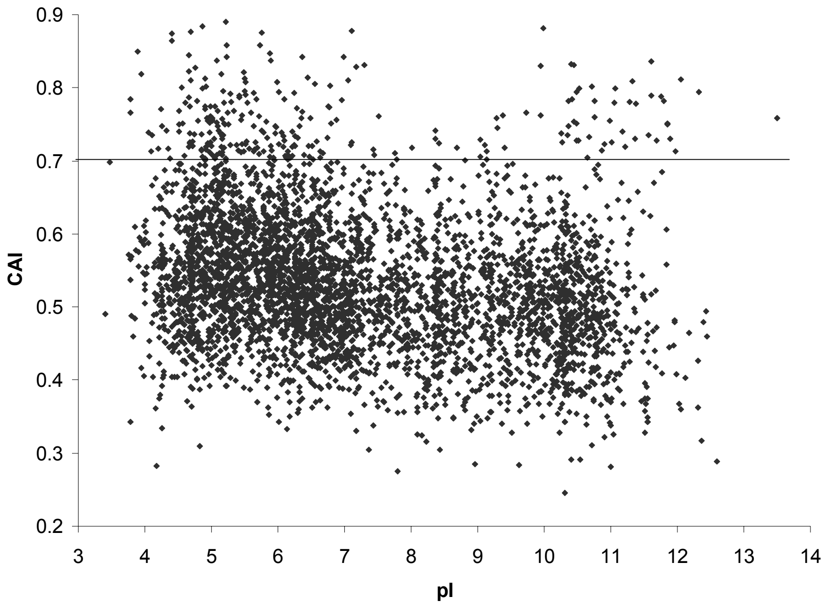


Figure 10-4. Few *E. coli* proteins with pI near environmental pH of 8-9 have high CAI values (an index of protein production). CAI and pI computed for all *E. coli* proteins longer than 33 amino acid residues, with the reference codon usage table Eeco_h.cut by using DAMBE (Xia, 2001; Xia and Xie, 2001b).

The data and reasoning above suggest that highly expressed proteins may indeed evolve (through their coding genes) to avoid having pI near the physiological pH of the organism as a mechanism to avoid forming protein aggregation and precipitation. This hypothesized precipitation-avoidance can explain the contrasting pI profiles in *E. coli* (Figure 10-1) and *H. pylori* (Figure 10-2). *E. coli* lives in mammalian intestine with a pH around 8-9. So we expect it to have few proteins with pI in the range of 8-9 because such proteins would tend to be uncharged, form aggregations and precipitate. For *H. pylori,* we expect it to have few proteins to have pI around 5 because that is its physiological (intracellular) pH when colonizing the mammalian stomach. This explains why the peak in the pH range 5-7 in *E. coli* (Figure 10-1) should be missing in *H. pylori* (Figure 10-2).

Now we have two valid hypotheses both of which can explain why *H. pylori* should have a basic set of proteins (or having its pI profile shifting to the right). Both hypotheses invoke selection, and interpret the basic set of proteins as an adaptation in response to selection in the acidic environment. However, the two hypotheses differ in what selection force is operating. To facilitate discussion, let us designate the adaptation hypothesis presented in the previous section as positive-shell hypothesis (PSH for short) and the adaptation hypothesis in this section as the precipitation-avoidance hypothesis (PAH for short).

In summary, PSH states that basic proteins in *H. pylori* are favored by selection because they foster the formation of a positively charged membrane to alleviate proton influx into the cell, whereas PAH argues that the basic proteins are favored (and acidic proteins selected against) to avoid precipitation. One may note that the two hypotheses are not mutually exclusive, and one may find supportive evidence for both hypotheses.

When the two hypotheses are expressed explicitly in this way, they clearly have different predictions. If PSH is correct, then it is the membrane proteins that should be affected most and they should spearhead the change in pI than the rest of the proteins. We have already investigated a small set of membrane proteins identified before (Baik *et al.*, 2004) in the previous section. We may subject all annotated proteins in *H. pylori* to bioinformatics tools, such as PSORT (Nakai and Horton, 1999), that reveal subcellular localization of proteins to identify other membrane proteins and check if it is generally true for membrane proteins to evolve towards an increase pI in *H. pylori* relative to its non-acid-resistance sister lineages.

In contrast, if PAH is correct, then it is those originally acidic and highly expressed proteins that are under the strongest selection pressure to evolve to have larger pI values. Thus, discriminating between these two hypotheses is reduced to test these two predictions. I consider it as a good idea to stop here and let the reader evaluate the two hypotheses.

Chapter 11

BIOINFORMATICS AND TWO-DIMENSIONAL PROTEIN SEPARATION

1. INTRODUCTION

In Chapter 10 we have briefly mentioned how to generate an expected pattern of proteins separated by their differences in isoelectric point and molecular mass, by assuming no posttranslational modification. This can then be compared with the actual separation pattern obtained by using 2D-SDS-PAGE which remains arguably the most frequently used proteomic method (Liebler *et al.*, 2002, p. 36). Such bioinformatics tools are valuable for identifying proteins that are the products of alternative splicing or that have undergone posttranslational modification.

Alternative splicing has now been recognized as the most fundamental mechanism in generating the complexity of multicellular eukaryotes. A limited number of protein-coding genes in multicellular eukaryotic genomes can generate a huge number of different proteins through alternative splicing (Ast, 2004). Even a single *Dscam* gene in *Drosophila melanogaster* can generate 38016 protein variants through alternative splicing (Graveley, 2005; Schmucker *et al.*, 2000). While alternative splicing is long known for generating the diversity of immunoglobulins, recent studies have shown that it affects the expression of many other genes (Kazan, 2003; Lipscombe, 2005; Stamm *et al.*, 2005), with an estimate of up to 60% of human genes (Kornblihtt, 2005; Lee and Wang, 2005) being involved in alternative splicing. Large-scale transcriptomic approaches such as microarray (Diehn *et al.*, 2000; Epstein and Butow, 2000; Gaasterland and Bekiranov, 2000; Holstege *et al.*, 1998; Schena, 1996; Schena, 2003) or SAGE (Madden *et al.*, 1997; Saha *et al.*, 2002; Velculescu *et al.*, 1995; Velculescu *et al.*, 1997; Zhang *et al.*, 1997) experiments are rather limited in detecting or predicting protein products resulting from alternative splicing, and direct proteomic methods are needed to characterize the diversity of protein products in multicellular eukaryotic cells.

Protein posttranslational modifications represent another biochemical mechanism contributing to the diversity of proteins in living cells. They typically involve changes in molecular mass or in charge. Such modified proteins will migrate to locations on the gel different from what we expect assuming no modification. Therefore, those proteins found at the same location in both the *in silico* gel and the real gel may be assumed to have undergone no posttranslational modification, whereas those whose coordinates do not match between the *in silico* gel and the real gel are good candidates for studying posttranslational modification.

The scientific rationale of the 2D-SDS-PAGE is outlined first in this chapter for readers not familiar with the method. This is followed by the simple computation needed to generate the *in silico* 2D-SDS-PAGE. We have already learned how to obtain the isoelectric point (pI) for each protein in the previous chapter. We only need to learn how to obtain the values for the other dimension (i.e., the molecular mass of the protein) and how to graphically present the results.

2. SCIENTIFIC RATIONALE BEHIND THE 2D-SDS-PAGE

2D-SDS-PAGE separate proteins by their differences in molecular mass and isoelectric point (pI). One may wonder why, given that proteins differ in many properties, should protein separation be based on their molecular mass and pI. Ideally, proteins should be separated on the basis of their properties least affected by the gel environment (e.g., loading buffer). Proteins have many different properties, such as molecular mass, charge and structure that can all affect protein mobility in the gel. Protein structures, in particular, are prone to environmental perturbations. In contrast, the molecular mass of a protein is stable and highly predictable as long as no active proteases are present in the gel. Similarly, each protein has its characteristic isoelectric point (i.e., the pH at which the protein does not carry any net charge, designated as pI), although it is not a protein property as rigid as the molecular mass. For example, even at a fixed pH, a certain amino group may become protonated at time t_1, deprotonated at time t_2, reprotonated at time t_3, and so on. However, the proportion of such amino groups, or the probability of such an amino group, being protonated remains the same for a given pH, leading to a relatively stable pI. For these reasons, protein separation by gel electrophoresis should be based on differences in molecular mass and pI, performed in a denaturing gel that reduces the proteins to linear structure. This is the scientific rationale behind the development of 2D-SDS-PAGE.

2D-SDS-PAGE first separate proteins based on their differences in pI by using immobilized pH gradient (IPG) strips that are available commercially. When loaded onto the strip under an electric field, proteins will migrate to the location of the strip with the pH equal to their pI values. At that location they do not carry any net charge and consequently will not move in the electric field. A protein wandering out of the location will carry charges again and will be forced back to the location by the electric field. This process, called isoelectric focusing (IEF), separate proteins along the pH gradient.

The second dimension of protein separation in 2D-SDS-PAGE is by molecular mass. Proteins are denatured by mercaptoethanol or DDT or the like. SDS then binds and imparts negative charge to proteins in roughly constant proportion to the length of the protein. This implies that each amino acid residue will be pulled roughly by the same electric force, and longer proteins, being clumsier in passing through porous materials, migrate more slowly than shorter ones. This resolves the protein mixture by molecular mass.

The DNA sequence in a living cell typically codes for many proteins. Prokaryotes typically have hundreds or thousands of protein-coding genes in their genome. The genome of the yeast, *Saccharomyces cerevisiae,* which is a unicellular eukaryote, contains about 6000 protein-coding genes. Human genome contains about 30000 protein-coding genes. However, the limited number of genes in the genome of multicellular eukaryotes can generate a huge number of different proteins through alternative splicing (Ast, 2004). Even a single *Dscam* gene in *Drosophila melanogaster* can generate 38016 protein variants through alternative splicing (Graveley, 2005; Schmucker *et al.*, 2000). Displaying and detecting these different proteins on a gel is by no means trivial. It is valuable for one to have an expected protein separation pattern as a reference to compare against the observed separation pattern on a 2D gel.

3. EXPECTED SEPARATION PATTERN OF 2D-SDS-PAGE FOR THE GENOME-DERIVED PROTEOME

For a well annotated genome, the protein-coding genes and their unmodified protein products are known. These annotated proteins are collectively known as the genome-derived proteome (gdProteome), in contrast to the conventional definition of a proteome as the collection of all quantifiable proteins in the same cell type at a particular developmental time. The latter is a cellular property at a specific time. The former, i.e., gdProteome, is a genomic property and does not change with cell type or time.

We have already learned how to compute the pI value of a known protein sequence. Here we learn how to compute the molecular mass of an amino acid sequence. It is important to know that the molecular mass of an amino acid sequence is not the summation of the molecular mass of all constituent amino acids, because a peptide is formed by the end-to-end condensation of amino acids with loss of water. The molecular mass of the resulting amino acid residues (Table 11-1) in a peptide, taken from Caltech's Protein/Peptide MicroAnalytical Laboratory at http://www.its.caltech.edu/~ppmal/, is consequently smaller than the molecular mass of intact amino acids by about 18 (i.e., molecular mass of H_2O).

Table 11-1. Molecular mass of amino acid residues. AA3 and AA1 refer to the 3-letter and 1-letter notation of amino acids.

AA3	AA1	AA Mass	Residue Mass
Gly	G	75.07	57.052
Ala	A	89.10	71.079
Ser	S	105.10	87.078
Pro	P	115.13	97.117
Val	V	117.15	99.133
Thr	T	119.12	101.105
Cys	C	121.20	103.144
Ile	I	131.17	113.160
Leu	L	131.18	113.160
Asn	N	132.10	114.104
Asp	D	133.10	115.089
Gln	Q	146.15	128.131
Lys	K	146.19	128.174
Glu	E	147.10	129.116
Met	M	149.21	131.198
His	H	155.16	137.142
Phe	F	165.19	147.177
Arg	R	174.20	156.188
Tyr	Y	181.19	163.170
Try	W	204.23	186.213

One may note that Ile and Leu have identical residue mass, and Gln and Lys have very similar residue mass. They cause problems in *de novo* sequencing of proteins by mass spectrometry (Carroll *et al.*, 2003). Bioinformatics tools to alleviate such problems will be presented in a latter chapter on proteomics.

The molecular mass of a peptide is the summation of the molecular mass of the constituent residues plus an extra proton (with a molecular mass of 1) at the N-terminal and an extra –OH (with a molecular mass of 17) at the C-terminal. Thus, the molecular mass of an amino acid sequence AACAGGRQD is 847.893.

The migration distance (D) can be expressed as a function of protein molecular mass (M). The following equation appears general enough to fit the relationship between D and M very well:

$$D = ae^{bM} \tag{11.1}$$

where a and b are constants that can be estimated by a simple linear regression of observed D on M values, after log-transformation of the two sides of the equation. Figure 11-1 shows the relationship between D and M, with a and b estimated to be 16.573 and -0.0283, respectively, based on my sample of a subset of secreted proteins of the gastric pathogen *Helicobacter pylori* (Bumann *et al.*, 2002).

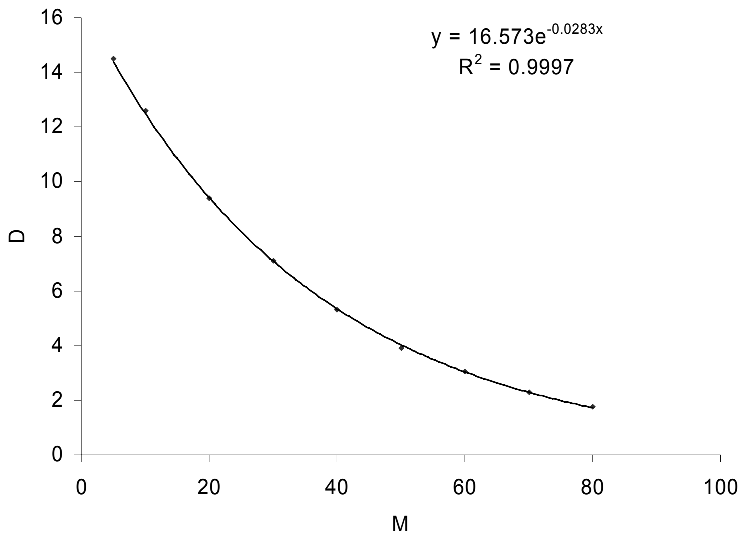


Figure 11-1. Relationship between migration distance (D) and molecular mass (M).

Now that we have protein pI, molecular mass and migration distance based on the molecular mass, the location of the protein on the 2D gel is defined. However, there is still one thing missing. A protein dot in a real 2D-

SDS-PAGE gel always features at least three types of information. That is, not only does it show the protein pI and the migration distance, it also reveals the protein abundance (i.e., big dots for highly expressed proteins and small dots for lowly expressed proteins). While we already know how to draw a protein dot on the 2D graph by using protein pI as the X-coordinate and D as the Y-coordinate, we need a measure of protein abundance so that different proteins will have different dot size. The *in silico* gel would not look professional if all dots were of the same size.

We know that highly expressed proteins exhibit strong codon usage bias. Thus, we can use an index of codon usage bias as an approximate measure of the dot size. This is not ideal, but is used for the lack of anything better. We have learned in Chapter 9 a number of indices for measuring codon usage bias, such as the effective number of codons (Wright, 1990), the codon adaptation index or CAI (Sharp and Li, 1987), the frequency of optimal codons or F_{op} (Ikemura, 1985), and the codon bias index or CBI (Bennetzen and Hall, 1982). Comparative studies (e.g.,Coghlan and Wolfe, 2000) suggest that CAI is the best in predicting gene expression levels. The *in silico* 2D-SDS-PAGE in Figure 11-2 is produced in this way by using CAI as a measure of protein abundance, i.e., the protein coding gene with a large CAI value will have a large dot. It includes all protein-coding genes longer than 100 in the budding yeast, *Saccharomyces cerevisiae.*

The *in silico* 2D-SDS-PAGE (Figure 11-2) reveals a set of highly expressed, positively charged, and relatively small proteins on the lower right of the gel. Many of these proteins contain a positively charged binding domain that may interact electrostatically with DNA and RNA (recall that DNA and RNA, with the phosphate backbone, are negatively charged and therefore exhibits affinity to positively charged proteins.

The protein pI values in *Saccharomyces cerevisiae* range from 3.2 to 13.1. This suggests that one should use an isoelectric focusing strip covering this range in a real 2D-SDS-PAGE. The *in silico* 2D-SDS-PAGE for *E. coli* exhibits a similar pattern with a number of small and positively charged proteins.

One may think that, if the yeast proteins already generate a rather crowded *in silico* 2D-SDS-PAGE, then it would be totally messy to display a much large number of possible proteins in a multicellular eukaryote. This concern seems unnecessary. Recent gene expression studies (e.g.,Velculescu *et al.*, 1999) have demonstrated that tissue-specific genes in multicellular organisms are relatively few. By taking advantage of such gene expression studies, we can produce tissue-specific *in silico* 2D-SDS-PAGE much simpler than that in Figure 11-2. Furthermore, eukaryotic proteins are distributed according to subcellular locations. For example, one can isolate ribosomes and study their 82 or so proteins (50 in the large ribosomal subunit and 32 in the small subunit).

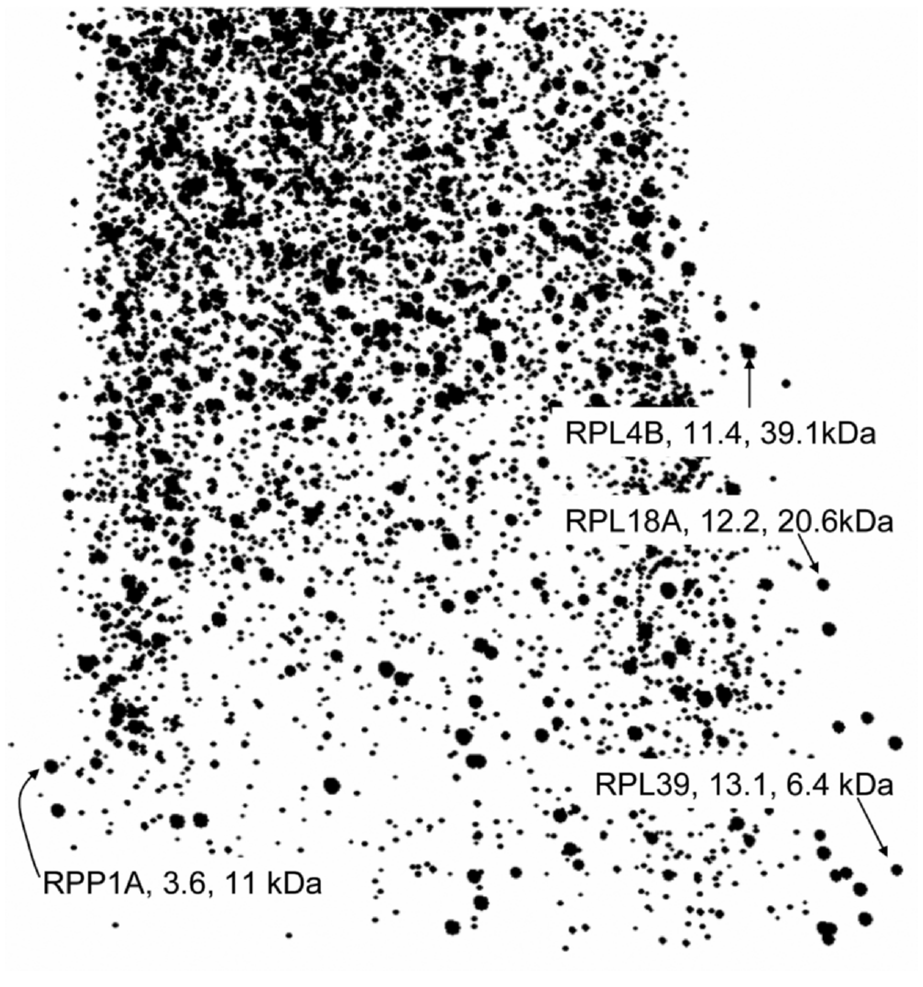


Figure 11-2. In silico 2D-SDS-PAGE of genome-derived proteins in the budding yeast, *Saccharomyces cerevisiae*, with four protein dots labeled. The annotations are in the form of "Gene name, pI, molecular mass". RPL4B, RPL18A and RPL39 are protein components of the large (60S) ribosomal subunit, all with high pI values and positively charged, which indicates the possibility of their electrostatic interaction with the negatively charged ribosomal RNA. RPP1A (acidic ribosomal protein P1α) is a component of the ribosomal stalk involved in the interaction between translational elongation factors and the ribosome. Produced with DAMBE (Xia, 2001; Xia and Xie, 2001b).

4. POSTTRANSLATIONAL MODIFICATION

4.1 Importance in studying posttranslational modification

While outstanding progresses have been made in characterizing transcriptomic data by microarray (Diehn *et al.*, 2000; Epstein and Butow, 2000; Gaasterland and Bekiranov, 2000; Holstege *et al.*, 1998; Schena, 1996; Schena, 2003) or SAGE (Madden *et al.*, 1997; Saha *et al.*, 2002; Velculescu *et al.*, 1995; Velculescu *et al.*, 1997; Zhang *et al.*, 1997) experiments, it is important to recognize the fact that an understanding of how living cells work ultimately depends on how well we understand the proteins in the cell.

Many proteins are not functional until they have undergone posttranslational modification. Take glucagon production for example. The gene coding for glucagon is transcribed and translated into proglucagon in both pancreas and intestine. However, proglucagon is cut to produce glucagon in the pancreas, but glucagon-like peptides in the intestine. The difference in glucagon production between the pancreas and the intestine is obvious at the protein level but not at the transcriptomic level.

A similar example is the differential production of different forms of somatostatin. Somatostatin SS-14 is secreted from pancreas, whereas SS-28 is the predominant form produced in the intestine. Both SS14 and SS-28 are derived from the same prosomatostatin which in turn is derived from preprosomatostatin. The difference in the production of SS-14 and SS-28 between the pancreas and the intestine is again obvious at the protein level but not at the transcriptomic level.

Posttranslational modification (PTM) of proteins plays a central role in protein activation and gene regulation. A fundamental and challenging aspect of proteomics is the identification of proteins that have undergone PTMs.

4.2 Posttranslational modification changes the migration pattern of proteins on 2D-SDS-PAGE

Nearly all PTMs result in a deviation of the protein dot from the expected *in silico* location. For this reason, an *in silico* 2D gel can facilitate the identification of PTM events.

A major class of PTMs involves addition of a function group that may alter both protein pI and molecular mass. For example, the amino group in the lysine is normally positively charged, but acetylation (the addition of an acetyl group, usually at the N-terminus of the protein or the lysine residue) results in the loss of the positive charge and a consequent decrease in pI (Figure 11-3). The modification also decreases migration distance because of the increased molecular mass. Thus, acetylation of amino acid residues in a protein will change the location of the affected protein on the gel.

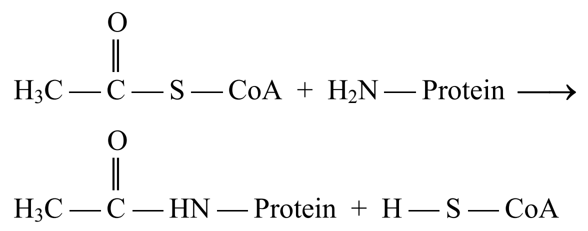

$H_3C—C(=O)—S—CoA + H_2N—Protein \longrightarrow$
$H_3C—C(=O)—HN—Protein + H—S—CoA$

Figure 11-3. Acetylation. Acetyl Coenzyme A (Acetyl CoA) attaches to the amino group of an amino acid residue (e.g., lysine, whose amino group typically exist in the form of —NH_3^+).

Acetylation plays a crucial role in gene expression. Eukaryotic genomic DNA typically wraps itself around a group of proteins in the nucleus called histones to form nucleosomes. Histones are rich in lysine residues and are consequently positively charged. These positively charged lysine residues interact electrostatically with the negatively charged DNA backbone to facilitate the formation of nucleosomes. RNA polymerases, which transcribe RNA from DNA, generally cannot access transcription start site when DNA is tightly wrapped around histones. Acetylation of the lysine residues removes the positive charge and allows the DNA to "melt" away from histones. This opens the transcription start site so that gene transcription can happen.

In vertebrates, when DNA is methylated at the 5-carbon of nucleotide C in CpG dinucleotides, these methylated CpG will attract methyl-CpG binding domain (MBD) of a number of proteins such as MBD1, MBD2, MBD3, and MeCP2. The resulting protein-DNA complex will recruit histone deacetylase which will remove acetyl-CoA from lysine residues to restore the positive charge of the amino group. These positively charged lysine residues then allow histones to bind tightly to DNA to prevent transcription.

Acetylation is observed only in eukaryotes. Aside from histones in the nucleus, about 59% and 90% of proteins in the cytoplasm are acetylated, mostly at the N-terminus. Proteins with its N-terminus acetylated are more resistant to degradation, with their lifetime extended from between seconds and hours up to days. Prokaryotic proteins, mitochondrial proteins and chloroplast proteins are not acetylated.

Another PTM that is unique in eukaryotes is glycosylation which involves the attachment of sugar molecules to specific amino acids in a polypeptide chain. This modification changes the molecular mass resulting in a change of the protein in the location of the 2D gel. Because small sugar molecules are water soluble, glycosylated proteins are also more soluble in water.

Glycosylation exists in two forms. The N-glycosylation involves the attachment of sugars to the nitrogen in the side chain of the amino acid asparagine (Asn). It requires a characteristic sequence of Asn-Xaa-Ser or Asn-Xaa-Thr where Xaa is any amino acid except proline. It is interesting that some N-glycosylation sites are never used, indicating that amino acids flanking the N-glycosylation sites might also be important. One can use the perceptron algorithm detailed in Chapter 5 to investigate this possibility, by collecting two sets of peptide sequences containing the N-glycosylation sites together with, say, 10 flanking amino acids. One set (the positive group) would include all those known to have undergone N-glycosylation in some tissue or at certain developmental stage, and the other set (the negative group) would include all those "N-glycosylation sites" that have never been N-glycosylated. Running the perceptron algorithm or its multilayer derivatives should shed light on the differences between the two groups.

The other form of glycosylation, called O-glycosylation, involves the attachment of sugars to the oxygen in the side chain of the amino acids serine or threonine. No specific sequence motif is associated with this form of glycosylation.

The most common and ubiquitous form of reversible protein modification is phosphorylation, which is crucial in signal transduction and enzyme regulation in the cell. It involves the addition of a phosphate (PO_4) group to the oxygen of the side chain of an amino acid residue, typically serine, threonine, and tyrosine residues. Such a modification would increase the negative charge of the protein, resulting in a downshift in pI. In addition, the involved protein will become more hydrophilic because of the charged (polar) phosphate group.

Reversible protein modifications such as phosphorylation typically involve a pair of enzymes working in opposite directions. Recall that acetylation and deacetylation require acetylase and deacetylase. Phosphorylation requires various protein kinases and dephosphorylation requires protein phosphatases.

Some PTMs involve the modification of certain function groups that may also change pI and molecular mass of the involved protein. For example, citrullination (or deimination which convert arginine to citrulline) results in the loss of charge in normally positively charged arginine residue, resulting in a downshift of protein pI.

Some PTMs involve the removal of certain function groups. For example, deamination removes an amide functional group from a chemical compound, and converts asparagine and glutamine residues to aspartic acid and glutamic acid residues. The modification reduces pI.

There are many other PTM events that occur frequently in a living cell and that can affect protein pI and migration distance. While great progress has been made in characterizing transcripts in living cells, very limited progress has been made to understand proteins. The true bottle neck in advancing our understanding of a living cell from a systems science point of view is proteomic analysis. It is for this reason that nearly all recently established institutions with an emphasis on systems biology feature strong proteomic expertise.

Chapter 12

SELF-ORGANIZING MAP AND OTHER CLUSTERING ALGORITHMS

1. INTRODUCTION

Self-organizing map (SOM) and other clustering algorithms have now become very popular in microarray data analysis. It might be beneficial to have an overview of clustering and classification methods used in bioinformatics before a numerical illustration of these algorithms. In particular, we need to understand some special terms in computational science such as classification and clustering, as well as supervised and unsupervised learning. Because clustering algorithms are frequently used in large-scale gene expression studies, especially in microarray experiments, I will also provide the scientific context in which the clustering algorithms are applied. In addition, because almost all clustering algorithms used in analyzing microarray data require a distance or a similarity index, we will also have a subsection on distances and similarities to help the reader in choosing which distance or similarity index to use. Finally we will numerically illustrate two algorithms, the UPGMA algorithm (Sneath, 1962) as a representative of the hierarchical clustering and the SOM algorithm (Kohonen, 2001) as a representative of the non-hierarchical clustering. These two algorithms are perhaps the most frequently used in analyzing microarray data.

1.1 Classification and clustering

Classification in bioinformatics and computer science is different from classification in taxonomy. Instead of referring to a method for organizing species into genera, families and higher taxa, classification in computer science refers to a class of algorithms that assign an entity with a set of measurable attributes (properties) to pre-defined groups, after the algorithm has already learned from a data set, termed training data, with many representative entities with known group identification. For example, one may characterize the expression of K genes for each of N liver cancer patients (Group 1) and M non-liver-cancer persons (Group 2) to obtain data in Figure 12-1 and apply a classification algorithm to learn the differences between the two groups. The result of learning from the training data can then be used to predict whether a person with measured properties E_1, E_2, …, E_k belongs to Group 1 or Group 2.

Group	E_1	E_2		E_k
1	$E_{1,1}$	$E_{1,2}$	…	$E_{1,K}$
1	$E_{2,1}$	$E_{2,2}$	…	$E_{2,K}$
…				
1	$E_{N,1}$	$E_{N,2}$	…	$E_{N,K}$
2	$E_{N+1,1}$	$E_{N+1,2}$	…	$E_{N+1,K}$
2	$E_{N+2,1}$	$E_{N+2,2}$	…	$E_{N+2,K}$
…				

Figure 12-1. Illustrative data layout for training two-group classification, with the first N rows (cases) belonging to Group 1 and the next M rows belonging to Group 2.

Both the single-layer perceptron and the two-group discriminant function analysis (better known as Fisher's linear discriminant function analysis or LDA) we covered in Chapter 5 are classification algorithms. In the case of perceptron, we may start with two groups of sequences (e.g., 10-base sequences flanking the initiation codon AUG from eukaryotes as the positive group and those from prokaryotes as the negative group) and the result of learning is a matrix that can be used to calculate a score to assign a new 10-base sequence to either the positive group (if the score is greater than 0) or the negative group (if the score is smaller than 0). Perceptron can work with either sequences or numerical data, but LDA works only with numerical data. Performing LDA with the type of data in Figure 12-1 results in a liner discriminant function that can be used to assign a person with measured properties E_1, E_2, …, E_k to Group 1 or Group 2.

People in computer sciences often talk about supervised learning and unsupervised learning. Supervised learning refers to the training process in which a classification algorithm derives the classification function from training data with known group identification. The training processes involving the perceptron algorithm and its multi-layer derivatives, LDA and its multi-group derivatives, support vector machine or SVM (Burges, 1998), etc., are all examples of supervised learning. In short, supervised learning is associated with classification into predefined groups.

What is then clustering? Clustering works with input data that do not have group identification, i.e., do not have the first column in Figure 12-1. Clustering algorithms are classified into hierarchical and non-hierarchical clustering algorithms. Representatives of the hierarchical clustering algorithms include conventional single-linkage, complete-linkage and average linkage algorithms, with the average linkage being used most often. The UPGMA (unweighted pair-group method with arithmetic mean) algorithm used during the early stage of molecular phylogenetics (Sneath, 1962; Sokal and Michener, 1958) is one of the average-linkage algorithms. Representatives of the non-hierarchical clustering algorithms include the K-mean (Hartigan, 1975) and self-organizing map or SOM (Kohonen, 2001). SOM is used extensively in analyzing microarray data (Chen *et al.*, 2001; Covell *et al.*, 2003; Kim *et al.*, 2005; Lamendola *et al.*, 2003; Ordway *et al.*, 2005; Seo *et al.*, 2005; Toronen *et al.*, 1999; Trutschl *et al.*, 2005; Wang *et al.*, 2002; Xiao *et al.*, 2003), and one may have difficulty understanding publications on microarray data without proper background knowledge of SOM.

1.2 Clustering and gene expression

One of the main objectives in gene expression studies is the identification of co-regulated genes and regulator-regulatee relationships. The first step in achieving the objective is to identify co-expressed genes. Clustering algorithms are used particularly often in identifying co-expressed genes (Xia and Xie, 2001a). Take human development for example. If we designate the time of zygote formation as t_0, what genes are activated at t_1, t_2, …, t_n? How do the products of these activated genes activate other genes and lead to the developmental cascade? If we know that Gene A activates Gene B which in turn activates Gene C, then we are in a good position to understand how living cells work. Note that "Gene X activates Gene Y" is understood to mean "the protein product of Gene X activates the transcription of Gene Y". If Genes A, B, and C are all activated by Gene D, then we know that Genes A, B, and C most likely share the same regulatory sequences controlled by the same transcription factor.

Microarray data publicly available are typically in the form of matrices each summarizing the expression profiles of thousands of gene loci either over a period of time or among different experimental conditions or cell types. Such data can provide two kinds of information that is related to transcription pathways and gene interactions. The first is the co-expressed genes whose expression may be controlled by the same gene product, e.g., their regulatory sequences may bind to the same transcription factor. The second is the regulator-regulatee relationship, in which one group of genes (regulatees) increase or decrease their expression consistently with the increase or decrease of the expression of another group of genes (regulators).

The co-expressed genes can be identified by calculating pair-wise similarity or dissimilarity indices among genes based on their expression profiles, and then clustered into gene clusters by using one of the many available clustering techniques (e.g., Bittner *et al.*, 1999; e.g., Chen *et al.*, 1999; Heyer *et al.*, 1999). Pearson correlation and the jackknife correlation (Heyer *et al.*, 1999) have been proposed. The former is not robust against outliers and the latter is too time-consuming to compute. An alternative is the nonparametric Spearman's r_S that is easy to compute and robust against one or multiple outliers.

For dissimilarity measures, the Euclidean distance has been suggested (Chen *et al.*, 1999; Heyer *et al.*, 1999). Other distances that can be used in clustering algorithms include Manhattan metric, percent remoteness, chord distance, and geodesic distance (Pielou, 1984). All these distances are metric (which is explained in the next section), and satisfy triangular inequality. My program AMIADA (Xia and Xie, 2001a), formerly called AMADA, implements these distances as well as the Pearson and Spearman correlations.

For clustering algorithms, both hierarchical and non-hierarchical ones have been proposed and used in research (Eisen *et al.*, 1998; Tamayo *et al.*, 1999; Tavazoie and Church, 1998; Tavazoie *et al.*, 1999; Wen *et al.*, 1998). The former include single-linkage, complete-linkage and average-linkage clustering (Pielou, 1984), and the latter include the k-mean clustering, the self-organization map, and the QT-clustering (Heyer *et al.*, 1999). The k-mean clustering requires the specification of the number of clusters (k) at the beginning, but k is unknown to the researcher. If the guessed k value is too large, then co-expressed genes may be split into different clusters. If the guessed value is too small, then unrelated genes may be forced into the same cluster. This problem is partially shared with the method of the self-organization map. The QT-clustering algorithm (Heyer *et al.*, 1999) has three disadvantages. First, it is time-consuming. Second, the criterion of choosing the cluster with the largest number of member as the best cluster is dubious. A more sensible criterion would be to choose a cluster as the best if it has the least overlap with others. Third, the "quality guarantee" is just a guessed value. I note that all clusters recovered by the QT-clustering (Heyer *et al.*, 1999) can also be recovered by the average-linkage method. So the former has no obvious advantage to offset the disadvantages. AMADA implements the single-linkage, complete-linkage and average-linkage algorithms.

Clustering aims to recover certain relationships (similarities) among the input entities (e.g., patients, genes) that do not have known group affiliation, in contrast to classification which works with data with known group affiliations (e.g., data in Figure 12-1). Hierarchical clustering is typically not associated with learning. Its ultimate objective is to build a hierarchical tree so that neighboring entities are more similar to each other than to those separated by more intermediate nodes. In other words, the generation of a tree is the end of the analysis and the algorithm does not keep any reusable knowledge for assigning data points not in the original data set to clusters on the tree.

In contrast, nonhierarchical clustering is associated with what is called unsupervised learning involving a training data set. Because the data points in the training set do not have group affiliation, the unsupervised learning process is supposed to learn a classification scheme from the training data and cluster the data points in the training data set to groups. However, clustering the data points is not the end of the analysis. The classification scheme (i.e., the knowledge) from the learning process will allow the classification of data points not in the original training data set. For example, the K-mean algorithm may cluster the training data into N groups and keep the centroids of these N groups. Once this is done (i.e., the training is over), a data point not in the original training set can be assigned to one of these N clusters by computing the distance between this data point and each of the N centroids, with the new data point classified into the group whose centroid has the smallest distance to the new data point. Similarly, once the training is finished for SOM, a new data point can be assigned to one of the nodes by checking which node is closest to the new data point.

We have previously learned that classification in machine-learning literature is associated with supervised learning. Now we know that nonhierarchical clustering is associated with unsupervised learning. I personally do not find it helpful to have terms or categorizations such as supervised and unsupervised learning because they do not seem to make algorithms easier to understand.

1.3 Similarity and distance indices

Almost all clustering algorithms used in analyzing microarray data require a distance or a similarity index. So we need to have basic understanding of distances and similarities. Representative distances that have been used in gene expression studies include the scale-dependent Euclidean distance (Bickel, 2003; Sawa and Ohno-Machado, 2003) and scale-independent Mahalanobis distance (Chilingaryan *et al.*, 2002) and (1-r) where r is Pearson correlation coefficient (Bickel, 2003; Eisen *et al.*, 1998; Sawa and Ohno-Machado, 2003). Other distances that could be used in clustering algorithms include Manhattan metric, percent remoteness, chord distance, and geodesic distance (Pielou, 1984). Mahalanobis distance becomes identical to Euclidean distance with standardized data, i.e., when variable X is transformed to x by

$$x_i = \frac{X_i - \overline{X}}{s_X} \tag{12.1}$$

so that mean and standard deviation of the resulting variable x is 0 and 1, respectively. Euclidean distance based on standardized data and (1-r) are perhaps used most frequently in clustering gene expression data. Representative similarity indices include various correlation coefficients such as the parametric Pearson correlation and non-parametric Spearman correlation and others.

An ideal distance or similarity index for clustering analysis should be metric. Nonmetric distances are likely to produce negative branch lengths in cluster analysis that are not only difficult to interpret, but also render frequently used optimization criteria (e.g., least-squares or minimum tree length) inapplicable.

So what is a distance in the first place? Designating x and y as two points in space, a distance is defined to have the following properties:

$D(x,x) = d_0$ (distance of a point to itself, which is typically 0)
$D(x,y) \geq d_0$
$D(x,y) = D(y,x)$

What is a metric distance? Designating x, y and z as three points in space, a metric distance is defined to have the following additional properties:

$D(x,y) = d_0$ if and only if $x = y$.
$D(x,z) \leq D(x,y) + D(y,z)$, or triangular inequality in Euclid geometry.

An example of a metric distance is the Euclidean distance. If d is a metric distance (e.g., Euclidean d), then the following are metric similarities:

$1/d$
Cd, where C is a constant
$d_{max} - d$

If s is a metric similarity index then the following are metric distances

$1/s$
Cs, where C is a constant
$s_{max} - s$

Pearson r is a similarity index, and its maximum is 1. Is (1- r) a metric distance? As (s_{max} – s) is a metric distance, (1 – r) would be a metric distance if r were a metric similarity. Because Euclidean distance is known to be metric, we can derive the relationship between Euclidean distance and r to help us conclude whether (1-r) is metric.

Although the formulation of Pearson r between variables x and y can be found in any statistical book, I have reproduced it below to facilitate presentation:

$$r_{xy} = \frac{\sum_{i=1}^{N}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{N}(x_i - \bar{x})^2 \sum_{i=1}^{N}(y_i - \bar{y})^2}} \tag{12.2}$$

To simplify the inference, we assume that the values of variable x and y have already been standardized so that the mean and standard deviation is 0 and 1, respectively. The variance of a standardized variable is also 1. This reduces Eq. (12.2) to

$$r_{xy} = \frac{\sum_{i=1}^{N} x_i y_i}{\sqrt{\sum_{i=1}^{N} x_i^2 \sum_{i=1}^{N} y_i^2}} = \frac{S_{xy}}{\sqrt{S_{xx}S_{yy}}} = \frac{S_{xy}}{\sqrt{S_{xx}^2}} = \frac{S_{xy}}{S_{xx}} = \frac{S_{xy}}{N-1} \tag{12.3}$$

The last step is often confusing to students because they have forgotten the fact that the variance of a standardized variable x (s^2) equals $S_{xx}/(N-1)$ and that $s^2 = 1$ for a standardized variable. Because $s^2 = 1 = S_{xx}/(N-1)$, $S_{xx} = N-1$. One should already know that $S_{xx} = S_{yy}$ for standardized variables x and y.

The Euclidian distance is defined as

$$d_{xy} = \sqrt{\sum_{i=1}^{N}(x_i - y_i)^2} = \sqrt{\sum_{i=1}^{N}(x_i^2 - 2x_i y_i + y_i^2)} \tag{12.4}$$

$$\begin{aligned} d_{xy}^2 &= \sum_{i=1}^{N} x_i^2 - 2\sum_{i=1}^{N} x_i y_i + \sum_{i=1}^{N} y_i^2 = S_{xx} - 2S_{xy} + S_{yy} \\ &= 2S_{xx} - 2S_{xy} = 2(S_{xx} - S_{xy}) \end{aligned} \tag{12.5}$$

We already know that S_{xx} = N-1. From Eq. (12.3), we also know that S_{xy} = (N-1)r_{xy}. This leads to

$$d_{xy}^2 = 2[(N-1) - (N-1)r_{xy}] = 2(N-1)(1-r_{xy}) \tag{12.6}$$

Because d^2_{xy} is not a metric distance, (1 – r) also is not a metric distance. In this sense, using Euclidean distance computed from standardized variables for clustering is better than using (1 – r). However, Euclidean distance is scale-dependent but (1 – r) is scale-independent. The scale effect is illustrated is Figure 12-2. We note that Gene 1 and Gene 3 increase and decrease in their expression synchronously, so do Gene 2 and Gene 4. In other words, Gene 1 and Gene 3 are co-expressed, so are Gene 2 and Gene 4. However because Gene 1 and Gene 2 are highly expressed and Gene 3 and Gene 4 are relatively lowly expressed, the synchronous change in gene expression between Gene 1 and Gene 3 and between Gene 2 and Gene 4 will not be reflected by Euclidean distance on the original data and the clustering analysis will not help us discover co-expressed genes.

	T0	T10	T20	T30	T40	T50	T60	T70
Gene 1	600	200	300	600	300	500	300	300
Gene 2	300	500	400	700	400	200	400	400
Gene 3	60	20	30	60	30	50	30	30
Gene 4	30	50	40	70	40	20	40	40

After Normalization

	T0	T10	T20	T30	T40	T50	T60	T70
Gene 1	1.369	-1.208	-0.564	1.369	-0.564	0.725	-0.564	-0.564
Gene 2	-0.772	0.600	-0.086	1.972	-0.086	-1.458	-0.086	-0.086
Gene 3	1.369	-1.208	-0.564	1.369	-0.564	0.725	-0.564	-0.564
Gene 4	-0.772	0.600	-0.086	1.972	-0.086	-1.458	-0.086	-0.086

Figure 12-2. Original data (top) and standardized data (bottom) to illustrate the effect of scale. T0, T10, etc., indicate time points.

You may note that, before standardization, Euclidean distance d_{12} between Gene 1 and Gene 2 is smaller than d_{13} or d_{14}, and d_{34} is the smallest of all pairwise distances. Application of a clustering algorithm typically will cluster Gene 3 and Gene 4 together, and then Gene 1 and Gene 2 together. In other words, the clustering analysis does not cluster co-expressed genes together.

After standardization (Figure 12-2, bottom half) which removes the scale effect, $d_{13} = d_{24} = 0$, which implies that Gene 1 should be clustered with Gene3 and Gene 2 should be clustered with Gene 4.

In contrast to Euclidean distance which differs between standardized and non-standardized data, the distance d' = (1-r) does not change with standardization, and $d'_{13} = d'_{24} = 0$ for either original or standardized data. Applying any cluster algorithm will results in Gene 1 and Gene 3 clustered together and Gene 2 and Gene 4 clustered together. If our purpose is to cluster co-expressed genes together, then using d' with either original or standardized data or using d with standardized data is better than using d with the original data.

Note that one does not have to standardize the data to remove the scale effect. For example, one can just transform the data so that all variables will have the same mean and variance.

2. UPGMA

We have learned two frequently used distance measures in the previous section, the Euclidean distance and (1 − r). Given N genes, there are N(N-1)/2 pairwise distances (designated as $d_{i,j}$ between gene i and gene j). We can apply the UPGMA algorithm to perform hierarchical clustering.

A matrix of $d_{i,j}$ values for five genes (designated as G_i) is shown in Figure 12-3. At this point we do not know the relationship among the genes and our ignorance is represented by what is called a star tree represented as (G_1,G_2,G_3,G_4,G_5). The UPGMA algorithm starts by finding the smallest $d_{i,j}$ (designated as d_{min}) in the matrix and cluster the two associated genes. There are two smallest $d_{i,j}$ in the matrix, $d_{1,3} = d_{4,5} = 0.0076$. So what should we do?

	G1	G2	G3	G4
G2	0.0916			
G3	0.0076	0.0840		
G4	0.0611	0.0611	0.0534	
G5	0.0534	0.0687	0.0458	0.0076

Figure 12-3. Distance matrix for illustrating UPGMA

In this particular example, we can start by either clustering G_1 and G_3 or clustering G_4 and G_5, and the final tree will be the same. However, identical

$d_{i,j}$ values sometime may lead to alternative clustering outcome. In particular, if $d_{i,j} = d_{i,k} = d_{min}$ (or $d_{i,j} = d_{j,k} = d_{min}$), then there will always be alternative trees because there is a clear conflict between clustering (G_i, G_j) and clustering (G_i, G_k). In other words, clustering G_i and G_j rules out the possibility of clustering G_i and G_k and vice versa. The neighbor-joining algorithm (Saitou and Nei, 1987) sometimes also experiences the problem of conflicting trees, although almost all phylogenetic programs implementing the UPGMA and neighbor-joining algorithms typically output only a single tree. While elegant algorithms are available to handle conflicts (Murtagh, 1984), the only computer program I know of that keeps track of conflicting trees from UPGMA and neighbor-joining method is DAMBE (Xia, 2001; Xia and Xie, 2001b).

Fortunately our simple example does not require us to resolve such a conflict. So we will proceed to cluster G_1 and G_3. This leads to a slightly more structured tree ((G_1,G_3),G_2,G_4,G_5) together with a reduced matrix shown in Figure 12-4.

	(G_1,G_3)	G_2	G_4
G_2	0.0878		
G_4	0.0573	0.0611	
G_5	0.0496	0.0687	0.0076

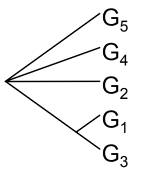


Figure 12-4. Intermediate result of UPGMA after clustering G_1 and G_3.

Note that some distances, e.g., $d_{2,4}$, $d_{2,5}$, $d_{4,5}$, in Figure 12-4 are directly transferred from the matrix in Figure 12-3. There are three new distances in the reduced matrix (Figure 12-4), computed as

$$\begin{aligned}
d_{2,(1,3)} &= \frac{d_{1,2} + d_{2,3}}{2} = \frac{0.0916 + 0.0840}{2} = 0.0878 \\
d_{4,(1,3)} &= \frac{d_{1,4} + d_{3,4}}{2} = \frac{0.0611 + 0.0534}{2} = 0.0573 \\
d_{5,(1,3)} &= \frac{d_{1,5} + d_{3,5}}{2} = 0.0496
\end{aligned} \qquad (12.7)$$

Now we again find d_{min} in the reduced matrix in Figure 12-4, which is d_{45} = 0.0076. So we cluster G_4 and G_5 and obtain a still more structured tree together with a still more reduced new matrix (Figure 12-5).

	(G_1,G_3)	G_2
G_2	0.0878	
(G_4,G_5)	0.0534	0.0649

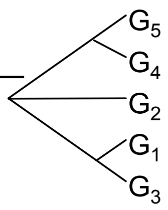


Figure 12-5. Intermediate result of UPGMA

There are two new distances in Figure 12-5, i.e.,

$$\begin{aligned} d_{(1,3),(4,5)} &= \frac{d_{(1,3),4} + d_{(1,3),5}}{2} = 0.0534 \\ d_{2,(4,5)} &= \frac{d_{2,4} + d_{2,5}}{2} = 0.0649 \end{aligned} \tag{12.8}$$

The smallest distance now is $d_{(1,3),(4,5)} = 0.0534$. This implies the clustering of (G_1,G_3) with (G_4,G_5). Now we have a fully resolved tree (Figure 12-6), together with the last distance computed as:

$$d_{((1,3),(4,5)),2} = \frac{d_{(1,3),2} + d_{(4,5),2}}{2} = 0.0764 \tag{12.9}$$

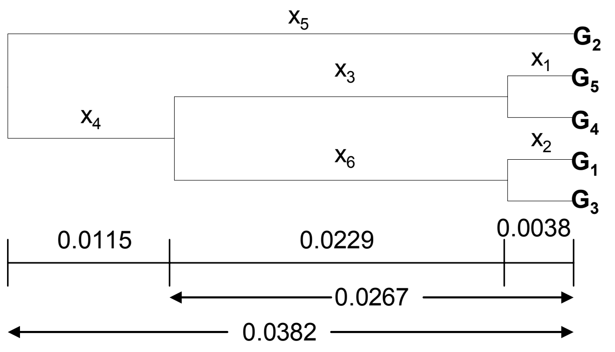


Figure 12-6. Final UPGMA tree.

The branch lengths of the tree can be easily computed. Because $d_{1,3} = d_{45} = 0.0076$, we have $x_1 = x_2 = d_{4,5}/2 = d_{1,3}/2 = 0.0038$, the branch length from G_1 or G_3 to their closest shared node is 0.0038. Similarly, $d_{(1,3),(4,5)} = 0.0534$, so we have $(x_1 + x_3) = (x_2 + x_6) = d_{(1,3),(4,5)}/2 = 0.0267$. Because $x_1 = 0.0038$, so $x_3 = 0.0267 - x1 = 0.0229$. Finally, because $d_{2,((1,3),(4,5))} = 0.0764$, so $x_5 = 0.0764/2 = 0.0382$. This also implies that $(x_1 + x_3 + x_4) = 0.0382$, so $x_4 = 0.0382 - x_1 - x_3 = 0.0115$. Also note that, in our example, $x_3 = x_6$. Figure 12-7 shows a more realistic output from a cluster analysis.

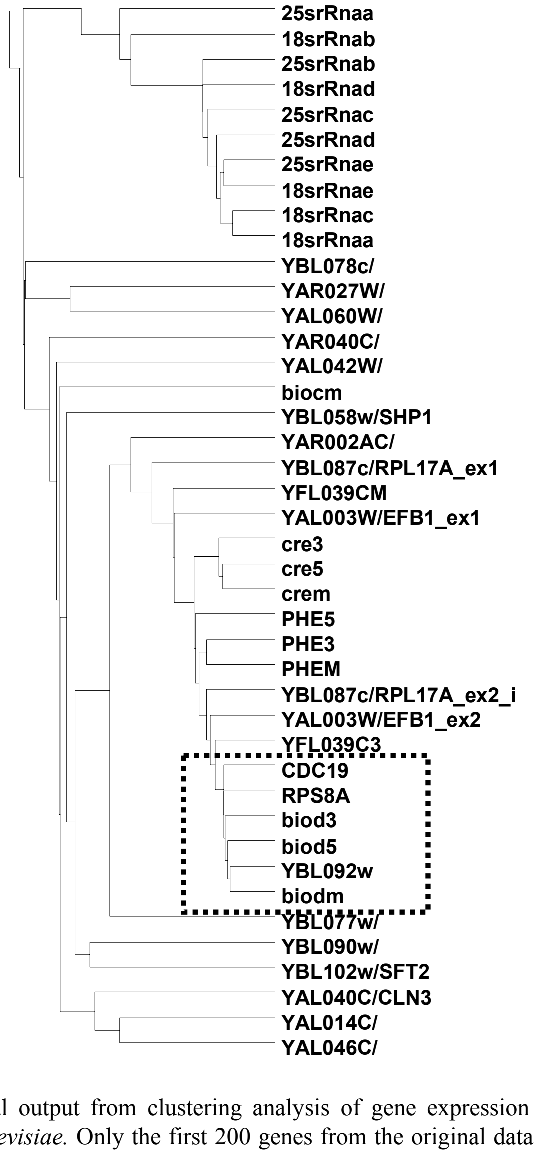


Figure 12-7. Partial output from clustering analysis of gene expression data in the yeast, *Saccharomyces cerevisiae.* Only the first 200 genes from the original data (Cho *et al.*, 1998) were used. The expression profiles of the six genes within the box of dashed lines are shown in Figure 12-8. Tree generated by AMIADA(Xia and Xie, 2001a).

Genes clustered together with short branch lengths connecting them should have similar expression profiles and are designated as co-expressed genes. I will illustrate the application of clustering analysis with a real example. The budding yeast, *Saccharomyces cerevisiae,* has a very useful property that the development of yeast cells in a culture can be synchronized. This allows one to monitor gene expression during the progression through the yeast cell cycle. In one of such studies (Cho *et al.*, 1998), 6220 transcripts were monitored and the data set is available to the public. Application of the UPGMA algorithm to the yeast gene expression data results in many clusters of co-expressed genes. Figure 12-7 displays a partial output of the clustering analysis using UPGMA and the standardized Euclidean distance for the first 200 genes.

My program AMIADA (Xia and Xie, 2001a) allows the user to easily visualize gene expression profiles. Right-clicking a node on the tree and choose 'Plot expression profiles' will display the expression profile of genes clustered under a node. Figure 12-8 shows a plot of the expression profiles for a set of six genes clustered together in Figure 12-7. The synchronized increase and decrease in their expression is obvious. Finding co-expressed genes is the first step towards identifying co-regulated genes, i.e., genes that share regulatory sequences.

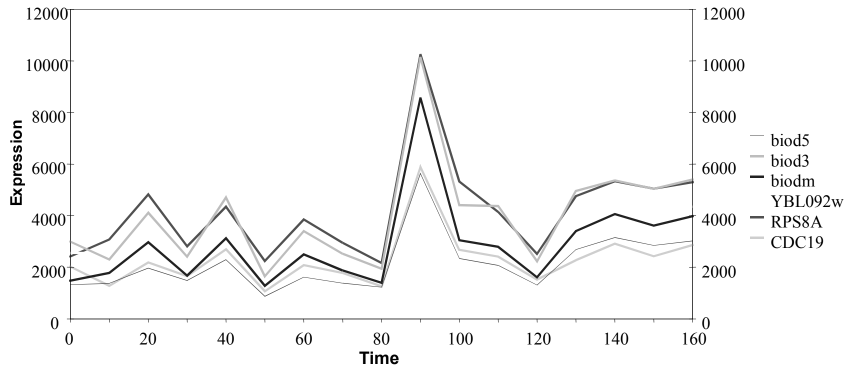


Figure 12-8. Co-expressed genes from yeast gene expression data (Cho *et al.*, 1998).

3. SELF-ORGANIZING MAP (SOM)

Being one of the unsupervised learning algorithms, SOM, like UPGMA, takes data that do not have prior group affiliation. It takes a training data set and goes through a training process to obtain a SOM of nodes (or artificial neurons) which can then be used for classification.

Because SOM is also one of the artificial neural network (ANN) algorithms, it is necessarily associated with concepts such as nodes (neurons) and learning rate. All ANN algorithms have neurons and need learning (training).

3.1 The SOM algorithm

We start with data in Table 12-1, where data are not standardized. If one is interested only in uncovering co-expressed genes, then one should standardize the data. The computation is the same regardless of our data being standardized or not.

Table 12-1. Fictitious gene expression data for illustrating the SOM algorithm. T0, T10 and T20 represent three time points. The values are between 0 and 100.

Gene	T0	T10	T20	Sum
1	93	76	87	256
2	80	81	85	246
3	89	88	85	262
4	69	74	96	239
5	95	89	93	277
6	65	96	76	237
7	87	85	96	268
8	78	89	88	255
9	87	80	97	264
10	67	96	55	218
11	91	90	95	276
12	76	72	67	215
13	79	78	94	251
14	96	76	78	250
15	66	64	63	193

Data in Table 12-1 is our training data. Most computation in SOM is in the training part. Once we have finished training, we will be able to use the finished SOM for classification of data points not in the training set.

We first need to decide the size of SOM, i.e., the number of nodes to have and whether the nodes should be arranged in one dimension, two dimensions, or higher dimensions. Most SOMs use a two-dimensional grid of nodes. There is not optimal way of choosing the number of nodes. Too many nodes increases computation and too few nodes may not provide sufficient fit to the data. For example, if we have only two nodes, then forcing all data points into these two nodes may result in very heterogeneous groupings. We will learn latter that a finished SOM provides some ways for us to see poorly fit points. If such points do exist, then we should re-run SOM with a larger grid of nodes.

Let us just start with a 3×3 grid of 9 nodes (Table 12-2), with randomly initialized values between 0 and 100 (the lower and upper bounds of our training data). Later on we will learn a better way of initialization. The random initialization symbolizes a beginning of ignorance. The knowledge will be gained through the learning process.

Table 12-2. A 3×3 grid of 9 nodes each with three randomly initialized values T0, T10 and T20.

	1			2			3		
	T0	T10	T20	T0	T10	T20	T0	T10	T20
1	2.2	11.5	33.4	41.9	27.6	0.8	6	63.8	51.2
2	28.5	30.2	47	28.7	51.3	9	61.6	38.2	17.9
3	40.5	76.1	71.2	79.8	94.6	23.2	76.2	40.9	23.9

We now randomly choose one gene, and suppose we happen to have chosen Gene 4 with T0, T10 and T20 equal to 69, 74 and 96, respectively (Table 12-1). The Euclidean distances (designated hereafter as d) between this gene and each of the 9 nodes (Table 12-3) show that Gene 4 is closest to node(3,1), with d = 37.8. This node is then called a winning node. You may use a distance other than the Euclidean, but the procedure is the same, i.e., you find the winning node which has the smallest distance to Gene 4.

Table 12-3. Euclidean distance between Gene 4 and each of the nine nodes.

	1	2	3
1	111.0	109.0	78.0
2	77.2	98.5	86.2
3	37.8	76.4	79.6

The winning node is the node that will get the first chance to learn from Gene 4, and its three values will be updated as a consequence of the learning. Recall that SOM is an algorithm in neural networks, and all neural networks learn. The updated values are given by the following equation referred to hereafter as a learning function:

$$w_i' = w_i(1-\alpha) + p_i\alpha \tag{12.10}$$

where w_1, w_2 and w_3 refers to the winning node's values in T0, T10 and T20, respectively, and p_1, p_2 and p_3 refers to the chosen gene's values in T0, T10 and T20, respectively. In our case, p_1, p_2 and p_3 equal 69, 74 and 96, respectively, and w_1, w_2 and w_3 equal 40.5, 76.1, and 71.2, respectively. Of course one can devise many alternative learning functions, but the one I have used is the simplest and it works fine. What is a bit tricky is the α parameter in Eq. (12.10), which is called the learning rate.

What should be the α value? An α equal to 0 implies $w_i' = w_i$, which means that the winning node does not learn anything from the chosen gene and will never change. An α equal to 1 implies $w_i = p_i$, which means that the winning node cannot retain any prior knowledge and will always mirror the knowledge of the chosen gene that has the smallest distance to it. This simple reasoning leads us to conclude that the α value should be greater than 0 but smaller than 1.

If α is close to 0, then the learning process is slow and the SOM takes a long time to converge. People with a very slow α often call themselves conservatives. If α is close to 1, the values of the winning node will change back and forth too fast and the node values may fail to stabilize. People with a α near 1 tend to call themselves liberals. Unfortunately, it would be very inconvenient to have two group of nodes, one with a small α and the other with a large α, fighting against each other to generate a winner. As a compromise, α will initially be large (close to 1), but will diminish with each iteration. This may make conservatives happy because all nodes in SOM will end up with a slow α and become conservatives.

For illustration, let's start with $\alpha = 0.5$ (Any value that is not an integer or half of an integer may cause headaches for a small fraction of biology students). This leads to

$$\begin{aligned} w_1' &= 40.5(1-0.5) + 69 \times 0.5 = 54.7 \\ w_2' &= 75.0 \\ w_3' &= 83.6 \end{aligned} \tag{12.11}$$

These three values will replace the three original values (i.e., 40.5, 76.1 and 71.2, respectively) of node(3,1). The update of the winning node is now complete.

The next step is to modify the neighbors of the winning node as the neighboring nodes will also learn from the chosen gene. This privilege of learning as a neighbor is what drives up the housing price near universities. Updating the values of neighbors is governed by the following learning function

$$w_i' = w_i(1-\alpha_n) + p_i\alpha_n \tag{12.12}$$

where α_n is the learning rate for the neighbors. Again one can use one of many possible alternative learning functions, but we will just use Eq. (12.12) to keep things simple. In practice, the learning function of neighbors depends on how we define neighbors and α_n will be larger for immediate neighbors than for remote neighbors. For obvious reasons, it should also be smaller than α. If we designate α_{n1} as the learning rate for the immediate neighbors (i.e., nodes in physical contact with the winning nodes), α_{n2} as the learning rate for the neighbors of the immediate neighbors, and so on, then one simple way of choosing α_n values is to set $\alpha_{n1} = \alpha/2$, $\alpha_{n2} = \alpha_{n1}/2$, and so on.

For our illustration, we will just define each node to have a maximum of four neighbors, i.e., the one to its left, the one to its right, the one above it and the one below it. Thus defined, we need only one α_n value which we will set to $\alpha/2$. Now a biology student typically will need a calculator to carry out the required arithmetic operations.

Our winning node is in a corner and consequently has only two neighbors, with one above it and one to its right. The updated values of SOM are shown in Table 12-4.

Table 12-4. SOM after updating the winning node, i.e., node(3,1), and its two neighbors, node(2,1) and node(3,2).

	1			2			3		
	T0	T10	T20	T0	T10	T20	T0	T10	T20
1	2.2	11.5	33.4	41.9	27.6	0.8	6.0	63.8	51.2
2	38.7	41.1	59.3	28.7	51.3	9.0	61.6	38.2	17.9
3	54.7	75.0	83.6	77.1	89.5	41.4	76.2	40.9	23.9

Now that we have done with Gene 4, we again repeat the process by randomly choosing a gene, computing the Euclidean distance to find the winning node, and carry out the updating of the values. We perform this with decreasing α and α_n values with each cycle of iteration until α equals a preset $\alpha_{min} > 0$ (We do not want to decrease α to zero because SOM will stop learning when $\alpha = 0$).

How should we decrease α and α_n with each cycle of updating? There is no optimal way of decreasing α. In my SOM implementation in AMIADA (Xia and Xie, 2001a), I used the following equation:

$$Q = (1 - \frac{1}{N_G}) \tag{12.13}$$

where N_G is the number of genes. In our case, $N_G = 15$ and $Q = 0.933$, i.e., α will be multiplied by Q after each cycle of iteration.

Continuing the learning process will eventually lead to convergence, i.e., when the values in the nodes do not change any more or the change is smaller than a pre-fixed small value in two consecutive cycles of iteration. The result of the learning process (Table 12-5) is a trained SOM ready for classification.

Table 12-5. Trained SOM.

	1			2			3		
	T0	T10	T20	T0	T10	T20	T0	T10	T20
1	80.7	77.6	73.3	74	77.9	67.3	69.2	79.5	72.2
2	84.4	81.9	82.1	82.5	81.2	81.4	78.5	77.6	78.2
3	89.6	85.4	91	88.1	82.9	91.7	79.9	79.1	89.4

Different nodes (Table 12-5) have different properties reflecting our data structure, e.g., we know that there are highly expressed genes and lowly expressed in the data set. The lower left nodes, i.e., node(3,1) and node(3,2) have relatively high expression values at all three time points, and node(1,2) and node(1,3) have relatively low expression at all three time points. Node(1,1) has its expression decreasing with time, node(3,3) has its expression increased at time T20 relative to the two previous time points.

The trained SOM can now be used for classification. During the classification stage, the node values do not change. A new gene with its expression values at T0, T10 and T20 can be assigned to a node by computing the Euclidean distance between this node and each of the nine nodes. The node with the smallest Euclidian distance will have the new gene assigned to it. The node to which Gene i is assigned is called the host node of Gene i.

Before we use the trained SOM to do the classification of new genes, it is crucial to check how well the SOM fits the training data. This is typically done by first assigning the genes in the training data to the nine nodes, and then computing the Euclidian distance (or its square) between each gene and its host node (Table 12-6).

Table 12-6. Classification of the 15 genes to the nine nodes. Row and Col indicate the coordinates of the host nodes. The last column is the Euclidian distance between each gene and its host node.

Gene	T0	T10	T20	Row	Col	d
1	93	76	87	3	2	9.69
2	80	81	85	2	2	4.41
3	89	88	85	3	1	6.54
4	69	74	96	3	3	13.77
5	95	89	93	3	1	6.80
6	65	96	76	1	3	17.43
7	87	85	96	3	2	4.90
8	78	89	88	3	3	10.17
9	87	80	97	3	2	6.12
10	67	96	55	1	2	23.00
11	91	90	95	3	1	6.30
12	76	72	67	1	2	6.19
13	79	78	94	3	3	4.84
14	96	76	78	2	1	13.63
15	66	64	63	1	2	16.54

We instantly notice a few genes that fit poorly into their respective host nodes (Table 12-6). For example, gene 10, with three gene expression values being 67, 96, and 55 at T0, T10 and T20, respectively, is classified to node(1,2) with its node values being 74, 77.9 and 67.3. Although both Gene 10 and its host node have the largest value at T10 and the smallest value at T20, the classification is deemed poor because of the large Euclidean distance (= 23, Table 12-6). The classification of Gene 7 to node(3,2) is similar, albeit with a smaller distance (= 17.43). Such large Euclidean distances suggest that the SOM does not provide good fit to the dada and we should re-run SOM with a larger (more accommodating) grid.

Application of SOM to the yeast gene expression data (Cho *et al.*, 1998) generates many co-expressed genes. Most are similar to those recovered by the UPGMA methods, but there are also different ones. These co-expressed genes are candidates for further study to check if they are co-regulated, i.e., whether they share similar regulatory sequences that are activated by the same transcription factors of similar proteins. The Gibbs sampler in Chapter 7 is one of the key data-mining tools used to identify such regulatory sequences.

It is important to recognize the fact that, although each input data point in our example is a vector of three numbers, SOM is not limited to data points represented as a vector of numbers. It can be applied to any data for which we can (1) define a distance between a data point and the node and (2) update the value of the winning nodes and neighboring nodes in response to the input. For example, the input data points can be 20-base sequences flanking the 5'-splicing site in eukaryotic protein-coding genes, and the node can be represented by a sequence profile (illustrated in Chapter 2 on profile alignment). The distance between the input sequence and the node sequence profile can just be a function of mismatch score or of an alignment score, and the updating of the nodes can be done easily by revising the sequence profile by adding the input sequence.

3.2 Variations of the basic SOM algorithm

I will briefly mention two variations of the basic SOM algorithm as presented in the previous section. The first is to replace the random initialization step by using the first two principal component scores from principal component analysis (PCA) of the matrix containing the gene expression data. PCA is a dimension reduction technique to project high dimensional data into a low dimensional space, and in this sense it serves a similar purpose as SOM.

To visualize the application of PCA to SOM initialization, plot the two principal component scores and superimpose the grid of node onto the two-dimensional plot (Figure 12-9). The values of each node, e.g., w_1, w_2 and w_3 in our example, are then the averages of those points falling within that node. The node values are then updated by using the same protocol as we have already learned. This dramatically reduces the computation time needed for SOM training.

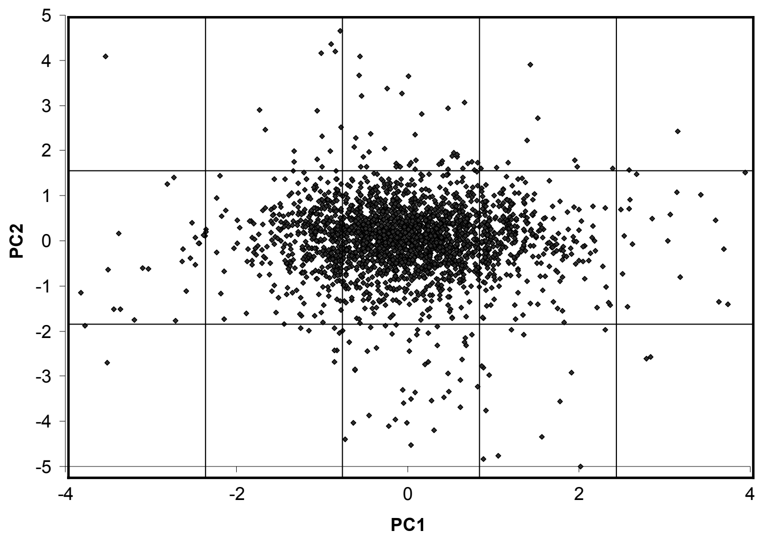


Figure 12-9. Using the first two principal component scores to initialize the grid of nodes.

One may wonder why the grid in Figure 12-9 has five columns but only three rows. The reason is that PC1 (the first principal component) typically accounts for far more variation in the data than PC2, and the dimension of the grid of nodes should be proportional to the variation accounted for by each principal component.

The other variation of the SOM algorithm involves the updating process. Instead of updating the winning node and its neighbors with every input data point, we simply find a host node for each data point and assign all data points to their respective host nodes without updating. Once all data points have been mapped onto the grid of nodes, we then compute the node values as the mean or median of those data points assigned to the node. This process is repeated until convergence is achieved. This variation of the SOM algorithm is often referred to as the batch mode of SOM training.

Chapter 13

MOLECULAR PHYLOGENETICS

1. INTRODUCTION

Molecular phylogenetics has been increasingly recognized as an essential subject in understanding molecular biology and evolution, and the subject has been often included in recent bioinformatics textbooks (Baxevanis and Ouellette, 2005; Higgs and Attwood, 2004). It is now a common consensus that understanding the dynamic nature of genes, genomes and gene interactions over evolutionary time is just as important as understanding the dynamic nature of gene expression and gene interaction during the development of an individual. We need molecular phylogenetics to facilitate our understanding of the dynamic natures of genes, genomes and gene interactions. In particular, phylogenetic relationships and dating are essential in reconstructing ancestral genes, predicting sites that are important to natural selection and, ultimately, understanding genomic evolution.

Four categories of phylogenetic methods are currently used to construct branching patterns during the speciation and gene duplication processes: the distance-based, the maximum parsimony, maximum likelihood and the Bayesian methods. Here I present the mathematical framework of the first three methods and their rationales, provide computational details for each of them, illustrate analytically and numerically the potential biases inherent in these methods, and outline computational challenges and unresolved problems. This is followed by a numerical illustration of the Bayesian inference, together with a few numerical illustrations of the Bayesian approach that has recently been used in molecular phylogenetics (Huelsenbeck *et al.*, 2001).

Molecular phylogenetics is now a very advanced subject in biology requiring substantial mathematical maturity. However, there are many excellent textbooks available (Felsenstein, 2004; Hillis *et al.*, 1996; Li, 1997; Nei and Kumar, 2000; Semple and Steel, 2003). This chapter, partially based on a previous review (Xia, 2007), will bridge the reader to more advanced topics in molecular phylogenetics. Although I have cut several corners to simplify and shorten the presentation, the chapter remains the longest in the book.

2. BIODIVERSITY, HISTORICAL INFORMATION, AND PHYLOGENETICS

Phylogenetics aims to organize biodiversity based on genealogical (ancestor-descendent) relationships. Biodiversity comes in many colors and shades, and unorganized biodiversity can not only dazzle our eyes but also confuse our minds. Molecular phylogenetics uses molecular sequence data to achieve its three main objectives: (1) to reconstruct the branching pattern of different evolutionary lineages such as species and genes, (2) to date evolutionary events such as speciation or gene duplication and subsequent functional divergence, and (3) to understand and summarize the evolutionary processes by substitution models. With the rapid increase of DNA and protein sequence data, and with the realization that DNA is the most reliable indicator of ancestor-descendent relationships, molecular phylogenetics has become one of the most dynamic fields in biology with solid theoretical foundations (Felsenstein, 2004; Li, 1997; Nei and Kumar, 2000; Page and Holmes, 1998; Semple and Steel, 2003) and powerful software tools (Felsenstein, 2002; Kumar *et al.*, 2001; Swofford, 2000; Xia, 2001; Xia and Xie, 2001b; Yang, 2002). I will not argue for the importance of molecular phylogenetics other than quoting Aristotle's statement that "He who sees things from the very beginning has the most advantageous view of them."

It is not always easy to see things from the very beginning. The evolutionary process depicted in Figure 13-1 shows an ancestral population with a single sequence shared among all individuals that have subsequently split into two populations and evolved and accumulated substitutions independently. Twelve substitutions have occurred, but only three differences can be observed between the sequences from the two extant species. The most fundamental difficulty in molecular phylogenetics is to estimate the true number of substitutions (i.e., 12) from the observed number of differences between extant sequences (i.e., 3). In short, the difficulty lies in how to correct for multiple hits.

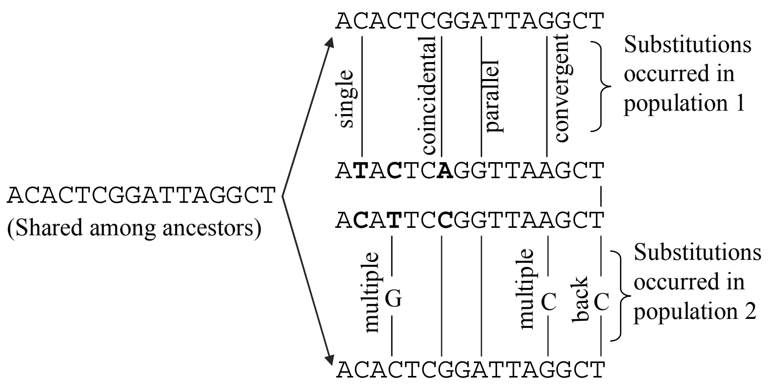


Figure 13-1. Illustration of nucleotide substitutions and the difficulty in correcting multiple hits. After Li (1997).

The number of substitutions per site is known as a genetic distance. The simplest genetic distance between two sequences, known as the p-distance (D_p), is simply the number of different sites (N) divided by the sequence length (L). For the two sequences in Figure 13-1, $D_p = 3/16$. Because D_p does not correct for multiple hits, it is typically a severe underestimate of the true genetic distance and has to be corrected.

In the next few sections, I will first detail commonly used substitution models, derive genetic distances based on the substitution models, and introduce the three categories of molecular phylogenetic methods: the distance-based, the maximum parsimony and the maximum likelihood methods. Several numerical examples are then presented to demonstrate commonly used computational approaches in Bayesian inference, such as conjugate prior distributions, discrete approximation and MCMC algorithms. Potential problems with these phylogenetic methods will be highlighted.

3. SUBSTITUTION MODELS

Substitution models reflect our understanding of how molecular sequences change over time. They are the theoretical foundation for computing the genetic distance in the distance-based phylogenetic method and for computing the likelihood value in the maximum likelihood method for phylogenetics. There are three types of molecular sequences, i.e., nucleotide, amino acid and codon sequences. Consequently, there are three types of substitution models, i.e., nucleotide-based, amino acid-based and codon-based. We will focus on nucleotide-based substitution models, with only a brief discussion on amino acid-based and codon-based models to highlight a few potential problems.

3.1 Nucleotide-based substitution models and genetic distances

Let $\boldsymbol{p_t}$ be the vector of the four nucleotide frequencies ($P_{A.t}$, $P_{G.t}$, $P_{C.t}$, $P_{T.t}$) at time t. Nucleotide-based substitution models are characterized by a Markov chain of four discrete states as follows:

$$p_{t+1} = p_t \begin{bmatrix} P_{AA} & P_{AG} & P_{AC} & P_{AT} \\ P_{GA} & P_{GG} & P_{GC} & P_{GT} \\ P_{CA} & P_{CG} & P_{CC} & P_{CT} \\ P_{TA} & P_{TG} & P_{TC} & P_{TT} \end{bmatrix} = p_t M \tag{13.1}$$

where $\boldsymbol{M}$ is the transition probability matrix and P_{ij} is the probability of changing from state i to state j in one unit of time. Three frequently used special cases of Eq. (13.1) will be detailed here: the JC69 model (Jukes and Cantor, 1969), the K80 model (Kimura, 1980), and the TN93 model (Tamura and Nei, 1993).

The simplest nucleotide substitution model is the JC69 one-parameter model, in which all off-diagonal elements in $\boldsymbol{M}$ are identical and designated as α. The four diagonal elements in $\boldsymbol{M}$ are 1-3α constrained by the row sum equal to 1. There is a corresponding rate matrix, designated by $\tilde{Q}$, that differs from $\boldsymbol{M}$ only in that the diagonal elements are -3α, constrained by the row sum equal to 0. It is often more convenient to derive substitution rates by using $\tilde{Q}$ instead of $\boldsymbol{M}$, as will be clear latter. Following Eq. (13.1), we have

$$\begin{aligned} P_{A.t+1} &= P_{A.t}(1-3\alpha) + P_{G.t}\alpha + P_{C.t}\alpha + P_{T.t}\alpha \\ P_{G.t+1} &= P_{A.t}\alpha + P_{G.t}(1-3\alpha) + P_{C.t}\alpha + P_{T.t}\alpha \\ P_{C.t+1} &= P_{A.t}\alpha + P_{G.t}\alpha + P_{C.t}(1-3\alpha) + P_{T.t}\alpha \\ P_{T.t+1} &= P_{A.t}\alpha + P_{G.t}\alpha + P_{C.t}\alpha + P_{T.t}(1-3\alpha)\,. \end{aligned} \tag{13.2}$$

Arranging the left side to be $P_{A.t+1}$ - $P_{A.t}$ and then applying the continuous approximation, we have

$$\begin{aligned}
\frac{\partial P_A}{\partial t} &= -P_{A.t}(3\alpha) + (P_{G.t} + P_{C.t} + P_{T.t})\alpha \\
\frac{\partial P_G}{\partial t} &= -P_{G.t}(3\alpha) + (P_{A.t} + P_{C.t} + P_{T.t})\alpha \\
\frac{\partial P_C}{\partial t} &= -P_{C.t}(3\alpha) + (P_{A.t} + P_{G.t} + P_{T.t})\alpha \\
\frac{\partial P_T}{\partial t} &= -P_{T.t}(3\alpha) + (P_{A.t} + P_{G.t} + P_{C.t})\alpha \ .
\end{aligned} \tag{13.3}$$

Equation (13.3) is a special case of a general equation. Designate ***d*** as the vector of the four partial derivatives, the general equation is

$$d = P_t\tilde{Q} \tag{13.4}$$

where $\tilde{Q}$ is the rate matrix mentioned before. The reason for $\tilde{Q}$ to be called a rate matrix should now be clear.

Suppose that we start with nucleotide A, what is the probability that it will stay as A or change to one of the other three nucleotides after time t? Given the initial condition that $P_{A.0} = 1$ and $P_{C.0} = P_{G.0} = P_{T.0} = 0$ and the constrain that that $P_A + P_G + P_C + P_T = 1$, Eq. (13.3) can be solved to yield

$$\begin{aligned}
P_{A.t} &= \frac{1}{4} + \frac{3}{4}e^{-4\alpha t} \\
P_{G.t} = P_{C.t} = P_{T.t} &= \frac{1}{4} - \frac{1}{4}e^{-4\alpha t} \ .
\end{aligned} \tag{13.5}$$

The time *t* in Eq. (13.5) is the time from the ancestor to the present. When we compare two extant sequences, the time is 2*t*, i.e., from one sequence to the ancestor and then back to the other sequence. So Eq. (13.5) has its general form as

$$\begin{aligned}
P_{ii.t} &= \frac{1}{4} + \frac{3}{4}e^{-8\alpha t} \\
P_{ij.t} &= \frac{1}{4} - \frac{1}{4}e^{-8\alpha t} \ .
\end{aligned} \tag{13.6}$$

The genetic distance (*D*), which is the number of substitutions per site, is defined as $2t\mu$ where μ is the rate of substitution. For the JC69 mode, $\mu = 3\alpha$, so $\alpha t = D/6$. Now we can readily derive *D* from the JC69 model, designated D_{JC69}, from the p-distance (D_p) defined before (Recall that D_p between two sequences is the probability of a site being different between two sequences). According to Eq. (13.6),

$$\begin{aligned} D_p &= 1 - P_{ii.t} = \frac{3}{4}(1 - e^{-8\alpha t}) = \frac{3}{4}(1 - e^{-4D_{JC69}/3}) \\ D_{JC69} &= -\frac{3}{4}\ln(1 - \frac{4D_p}{3}) \,. \end{aligned} \tag{13.7}$$

For the two sequences in Figure 13-1, D_p = 3/16 = 0.1875 and D_{JC69} = 0.21576. The equilibrium frequencies are derived by setting ($p_{i.t+1} - p_{i.t}$) in Eq. (13.3) to zero. Solving the resulting simultaneous equations with the constraint that the four frequencies sum up to 1, we have $P_{A.t} = P_{G.t} = P_{C.t} = P_{T.t}$ = 0.25. In summary, the JC69 model assumes that (1) the four nucleotides can change into each other with equal probability and (2) the equilibrium frequencies are all equal to 0.25.

The variance of D_{JC69} can be obtained by using the “delta” method (Kimura and Ohta, 1972). When a variable Y is a function of a variable X, i.e., Y = F(X), the delta method allows us to obtain approximate formulation of the variance of Y if (1) Y is differentiable with respect to X and (2) the variance of X is known. The same can be extended to more variables.

The mathematical concept for the delta method is illustrated below, starting with the simplest case of Y = F(X). Regardless of the functional relationship between Y and X, we always have

$$\Delta Y \approx \left(\frac{dY}{dX}\right)\Delta X \tag{13.8}$$

$$(\Delta Y)^2 \approx \left(\frac{dY}{dX}\right)^2 (\Delta X)^2 \,. \tag{13.9}$$

where ΔY and ΔX are small changes in Y and X, respectively.

Note that the variance of Y is the expectation of the squared deviations of Y, i.e.,

$$\begin{aligned} V(Y) &= E(\Delta Y)^2 \\ V(X) &= E(\Delta X)^2 \,. \end{aligned} \tag{13.10}$$

Replacing $(\Delta Y)^2$ and $(\Delta X)^2$ in Eq. (13.9) with $V(Y)$ and $V(X)$, we have

$$V(Y) \approx \left(\frac{dY}{dX} \right)^2 V(X) \, . \tag{13.11}$$

This relationship allows us to obtain an approximate formulation of the variance of either *Y* or *X* if we know either *V*(*X*) or *V*(*Y*). For the variance of D_{JC69}, we note that D_{JC69} is a function of D_p, and the variance of D_p is known from the binomial distribution:

$$V(D_p) = \frac{D_p(1 - D_p)}{L} \tag{13.12}$$

where L is the length of the two aligned sequences. From the expression of D_{JC69} in Eq. (13.7), we have

$$\begin{aligned} \frac{\partial D_{JC69}}{\partial D_p} &= \frac{1}{1 - \frac{4D_p}{3}} \\ V(D_{JC69}) &= \left(\frac{\partial D_{JC69}}{\partial D_p} \right)^2 V(D_p) = \frac{D_p(1 - D_p)}{L\left(1 - \frac{4D_p}{3}\right)^2} \, . \end{aligned} \tag{13.13}$$

As students are always eager to have more illustrative examples, and because I myself belong to the lesser folks who cannot see the beauty of equations without rendering them to numbers, I will present another example, taken from a book on population genetics (Li, 1976), in which we know the variance of Y and want to estimate the variance of X.

Given a locus with one dominant allele (A) and one recessive allele (a), we have only two distinguishable phenotypes, dominants (AA, Aa) and recessives (aa). How to estimate the allele frequency of a and its variance?

You might be interested to know that the severe human disease cystic fibrosis is determined by one locus with a dominant allele and a recessive allele. The disease is caused by homozygosity for the recessive allele.

Let D and R be the observed numbers of dominants and recessives in a sample of N random individuals (N = D + R). Our estimate of the frequency of allele a, designated q, is

$$\begin{aligned} q^2 &= R / N \\ q &= \sqrt{R / N} \end{aligned} \tag{13.14}$$

In the case of cystic fibrosis, the ratio of R/N is about 1/2500. So q = 1/50. Now we proceed to find the variance of q. From the binomial distribution, we know the variance of q^2 to be

$$V(q^2) = \frac{q^2(1-q^2)}{N} \tag{13.15}$$

In the framework of the delta method with Y = F(X), we have Y = q^2, X = q, and dY/dX = 2q. We already know the variance of Y (i.e., q^2) in Eq. (13.15), and the variance of X can be obtained as follows

$$\begin{aligned} V(Y) &= \frac{q^2(1-q^2)}{N} = \left(\frac{dY}{dq}\right)^2 V(q) = (2q)^2 V(q) \\ V(q) &= \frac{V(Y)}{(2q)^2} = \frac{\frac{q^2(1-q^2)}{N}}{4q^2} = \frac{1-q^2}{4N} \end{aligned} \tag{13.16}$$

You might have noticed that, q^2 = R/N and (1-q^2) = D/N. So we have

$$V(q) = \frac{1-q^2}{4N} = \frac{D/N}{4N} = \frac{D}{4N^2} \tag{13.17}$$

In the case of cystic fibrosis, q = 1/50 = 0.02, V(q) = 0.00009996, the standard deviation of q is 0.009998, and the 95% confidence interval for q, when sample size is large, is (0.0004, 0.0396).

Joe Felsenstein (pers. comm.) suggests that a bioinformatics book would be incomplete without coverage of the EM algorithm. So here comes a numerical illustration (the simplest possible, I believe) of the EM algorithm to estimate q (the frequency of the recessive allele a). Note that in this particular case we do not need to use the EM algorithm to estimate q because q is already given in Eq. (13.14). However, gaining some familiarity with the EM algorithm may be useful in other situations.

There are three genotypes involving the cystic fibrosis locus, AA, Aa and aa, with AA and Aa indistinguishable phenotypically. Designate p = 1 – q and let N_{AA}, N_{Aa} and N_{aa} be the number of AA, Aa and aa genotypes, respectively. Using the notations above, we have D = (N_{AA} + N_{Aa}) and R = N_{aa}. Because the EM algorithm works on real data, we will assume that we have observed D = 9996 and R = 4.

Since we cannot observe N_{AA} and N_{Aa} directly (they are indistinguishable phenotypically), the two numbers represent incomplete data from a three-category trinomial distribution. The complete data specification is as follows:

$$
\begin{aligned}
&f(N_{AA}, N_{Aa}, N_{aa} \mid q) = C[p^2]^{N_{AA}}[2pq]^{N_{Aa}}[q^2]^{N_{aa}} \\
&C = \frac{N!}{N_{AA}!N_{Aa}!N_{aa}!}
\end{aligned}
\tag{13.18}
$$

The EM algorithm consists of two steps, the estimation step (or E-step) and the maximization step (or M-step). Let us start by setting q = 0.1. For the E-step, we estimate N_{AA} and N_{Aa} as follows (with the subscript in $N_{AA.1}$ and $N_{Aa.1}$ indicating the first E-step):

$$
\begin{aligned}
&N_{AA.1} = D\frac{p^2}{p^2 + 2pq} = 8178.545455 \\
&N_{Aa.1} = D\frac{2pq}{p^2 + 2pq} = 1817.454545
\end{aligned}
\tag{13.19}
$$

Substituting these into Eq. (13.18), we can obtain q by the maximum likelihood method, i.e., taking the derivative of $f(N_{AA},N_{Aa},N_{aa}|q)$ with respect to q, setting the derivative to 0 and solve the resulting equation for q. This gives

$$
q_1 = \frac{N_{Aa} + 2N_{aa}}{2N} = 0.091272727 \tag{13.20}
$$

where the subscript 1 in q_1 indicates the first M-step. We now repeat the E-step according to the following equations equivalent to Eq. (13.19)

$$
\begin{aligned}
&N_{AA.i} = D\frac{p^2}{p^2 + 2pq_{i-1}} \\
&N_{Aa.i} = D\frac{2pq_{i-1}}{p^2 + 2pq_{i-1}} \\
&p = 1 - q_{i-1}
\end{aligned}
\tag{13.21}
$$

and the M-step according to the following equation equivalent to Eq. (13.20)

$$q_i = \frac{N_{Aa.i} + 2N_{aa}}{2N} \tag{13.22}$$

Repeating the E-step and M-step will result in q_i asymptotically approaching 0.02, and N_{Aa} and N_{AA} approaching 392 and 9604, respectively.

Let us now come back to molecular phylogenetics and introduce a slightly more complicated substitution model. Kimura (1980) noted that transitional substitutions typically occur much more frequently than transversions, and consequently proposed the two-parameter K80 model in which the rate of transitional substitutions (A↔G and T↔C) is designated as α and the rate of transversion substitutions (A↔T, A↔C, G↔T and G↔C) as β:

$$\tilde{Q} = \begin{bmatrix} A & \bullet & \alpha & \beta & \beta \\ G & \alpha & \bullet & \beta & \beta \\ C & \beta & \beta & \bullet & \alpha \\ T & \beta & \beta & \alpha & \bullet \end{bmatrix}. \tag{13.23}$$

Substituting this new $\tilde{Q}$ into Eq. (13.4) and solving the equations with the initial condition that $P_{A.0} = 1$ and $P_{C.0} = P_{G.0} = P_{T.0} = 0$ and the constrain that that $P_A + P_G + P_C + P_T = 1$ as before, we have

$$\begin{aligned} P_{A.t} &= \frac{1}{4} + \frac{1}{4}e^{-4\beta t} + \frac{1}{2}e^{-2(\alpha+\beta)t} \\ P_{G.t} &= \frac{1}{4} + \frac{1}{4}e^{-4\beta t} - \frac{1}{2}e^{-2(\alpha+\beta)t} \\ P_{C.t} = P_{T.t} &= \frac{1}{4} - \frac{1}{4}e^{-4\beta t}\ . \end{aligned} \tag{13.24}$$

Note again that time t in Eq. (13.24) should be $2t$ when used between two extant sequences. So Eq. (13.24) has its general form as

$$\begin{aligned} P_{s.t} &= \frac{1}{4} + \frac{1}{4}e^{-8\beta t} - \frac{1}{2}e^{-4(\alpha+\beta)t} \\ P_{v.t} &= \frac{1}{2} - \frac{1}{2}e^{-8\beta t} \end{aligned} \tag{13.25}$$

where $P_{s.t}$ and $P_{v.t}$ are the probabilities that a site differs by a transition and a transversion, respectively, between two sequences that have diverged for time t, and can be estimated by the proportion of sites differing by a transition (P) and a transversion (Q), respectively. This leads to

$$\begin{aligned} \beta t &= -\frac{\ln(1-2Q)}{8} \\ \alpha t &= -\frac{\ln(1-2P-Q)}{4} + \frac{\ln(1-2Q)}{8} \ . \end{aligned} \qquad (13.26)$$

Recall that the genetic distance is defined as $2t\mu$ where $\mu = \alpha + 2\beta$ for the K80 model. Therefore,

$$\begin{aligned} &D_{K80} = 2\alpha t + 4\beta t = \frac{1}{2}\ln(a) + \frac{1}{4}\ln(b), \text{ where} \\ &a = \frac{1}{1\text{-}2P\text{-}Q} \text{ and } b = \frac{1}{1\text{-}2Q} \ . \end{aligned} \qquad (13.27)$$

For the two sequences in Figure 13-1, P = 2/16, Q = 1/16, D_{K80} = 0.22073. The equilibrium frequencies are derived by setting $\boldsymbol{d}$ in Eq. (13.4) to the $\boldsymbol{0}$ vector. Solving the resulting simultaneous equations with the constraint that the four frequencies sum up to 1, we have $P_{A.t} = P_{G.t} = P_{C.t} = P_{T.t}$ = 0.25. Thus, the K80 model shares with the JC69 model the assumption that the equilibrium frequencies are all equal to 0.25. You might have noticed this because nucleotide frequencies are not featured in the expression of D_{JC69} or D_{K80}.

The variance of D_{K80} can be derived by the delta method as before:

$$dD_{K80} = \left(\frac{\partial D_{K80}}{\partial P}\right)dP + \left(\frac{\partial D_{K80}}{\partial Q}\right)dQ = d_1 \bullet dP + d_2 \bullet dQ \qquad (13.28)$$

$$\begin{aligned}
V(D_{K80}) &= (dD_{K80})^2 = [d_1 \bullet dP + d_2 \bullet dQ]^2 \\
&= d_1^2 dP^2 + 2d_1 d_2 dPdQ + d_2^2 dQ^2 \\
&= d_1^2 V(P) + 2d_1 d_2 Cov(P,Q) + d_2^2 V(Q) \\
&= \begin{bmatrix} d_1 & d_2 \end{bmatrix} \begin{bmatrix} V(P) & Cov(P,Q) \\ Cov(P,Q) & V(Q) \end{bmatrix} \begin{bmatrix} d_1 \\ d_2 \end{bmatrix}.
\end{aligned} \tag{13.29}$$

Recall that P stands for the proportion of sites that differ by a transitional change and Q stands for the proportion of sites that differ by a transversional change. Designate R as the proportion of identical sites ($R = 1 - P - Q$). From the trinomial distribution of $(R + P + Q)^L$, we have:

$$\begin{aligned}
V(P) &= \frac{P(1-P)}{L} \\
V(Q) &= \frac{Q(1-Q)}{L} \\
Cov(P,Q) &= -\frac{PQ}{L}\ .
\end{aligned} \tag{13.30}$$

Substituting these into Eq. (13.29), we have the variance of D_{K80}:

$$V(D_{K80}) = (dD_{K80})^2 = \frac{a^2 P + c^2 Q - (aP + cQ)^2}{L} \tag{13.31}$$

where $c = (a + b)/2$, with a and b defined in Eq. (13.27).

Note that Eq. (13.29) is a general equation for computing the variance by the delta method. For any function $Y = \mathrm{F}(X_1, X_2, ..., X_n)$, the variance of Y is obtained by the variance-covariance matrix of X_i multiplied left and right by the vector of partial derivatives of Y with respect to X_i.

Tamura and Nei (1993) noticed the rate difference between C↔T and A↔G transitions and proposed the TN93 model with the following rate matrix:

$$\tilde{Q} = \begin{bmatrix} A & \bullet & \alpha_2\pi_G & \beta\pi_C & \beta\pi_T \\ G & \alpha_2\pi_A & \bullet & \beta\pi_C & \beta\pi_T \\ C & \beta\pi_A & \beta\pi_G & \bullet & \alpha_1\pi_T \\ T & \beta\pi_A & \beta\pi_G & \alpha_1\pi_C & \bullet \end{bmatrix}. \tag{13.32}$$

where π_i designates equilibrium nucleotide frequencies, and the diagonal is constrained by the row sum equal to 0.

Following the same protocol as before, and designate P_1, P_2 and Q as the probabilities of C↔T transitions, A↔G transitions and R↔Y transversions (R means either A or G and Y means either C or T), respectively, we can obtain,

$$P_1 = \pi_T P_{TC}(2t) + \pi_C P_{CT}(2t) = \frac{2\pi_T\pi_C(\pi_Y + \pi_R e^{-2\beta t} - e^{-2(\alpha_1\pi_Y + \beta\pi_R)t})}{\pi_Y} \tag{13.33}$$

$$P_2 = \pi_A P_{AG}(2t) + \pi_G P_{GA}(2t) = \frac{2\pi_A\pi_G(\pi_R + \pi_Y e^{-2\beta t} - e^{-2(\alpha_2\pi_R + \beta\pi_Y)t})}{\pi_R} \tag{13.34}$$

$$Q = 2\pi_R\pi_Y(1 - e^{-2\beta t}) \ . \tag{13.35}$$

Solving for $\alpha_1 t$, $\alpha_2 t$ and βt from Eqs. (13.33)-(13.35), we have

$$\alpha_1 t = \frac{-\ln(1 - \frac{Q}{2\pi_Y} - \frac{P_1\pi_Y}{2\pi_T\pi_C}) + \pi_R \ln(1 - \frac{Q}{2\pi_R\pi_Y})}{2\pi_Y} \tag{13.36}$$

$$\alpha_2 t = \frac{-\ln(1 - \frac{Q}{2\pi_R} - \frac{P_2\pi_R}{2\pi_A\pi_G}) + \pi_Y \ln(1 - \frac{Q}{2\pi_R\pi_Y})}{2\pi_R} \tag{13.37}$$

$$\beta t = -\frac{\ln(1 - \frac{Q}{2\pi_R\pi_Y})}{2} \ . \tag{13.38}$$

$$\begin{aligned} D_{TN93} &= 2t[\pi_A(\beta\pi_T + \beta\pi_C + \alpha_2\pi_G) + \pi_C(\beta\pi_A + \beta\pi_G + \alpha_1\pi_T) \\ &\quad + \pi_T(\beta\pi_A + \beta\pi_G + \alpha_1\pi_C) + \pi_G(\beta\pi_T + \beta\pi_C + \alpha_2\pi_A)] \\ &= 4[\pi_R\pi_Y\beta t + \pi_A\pi_G\alpha_2 t + \pi_C\pi_T\alpha_1 t] \ . \end{aligned} \tag{13.39}$$

Because we can estimate P_1, P_2 and Q by the proportion of sites with C↔T transitions, A↔G transitions and R↔Y transversions, respectively, D_{TN93} can be readily computed. For the two sequences in Figure 13-1, D_{TN93} is 0.2525. The variance of D_{TN93} can be easily obtained by left- and right-multiplying the variance-covariance matrix of P_1, P_2 and Q with the vector of the three derivatives of D_{TN93} with respect to P_1, P_2 and Q in the same way shown in the last term of Eq. (13.29). The variance and covariance of P_1, P_2 and Q can be obtained in the same way as in Eq. (13.30).

Many more substitution models and genetic distances have been proposed (Tamura and Kumar, 2002), with the number of all possible time reversible models of nucleotide substitution being 203 (Huelsenbeck *et al.*, 2004). In addition, there are more complicated models underlying the LogDet and the paralinear distances (Lake, 1994; Lockhart *et al.*, 1994) that can presumably accommodate the nonstationarity of the substitution process. Such models have not been implemented in a maximum likelihood framework until very recently (Jayaswal *et al.*, 2005). Different substitution models often lead to different trees produced and constitute a major source of controversy in molecular phylogenetics (Rosenberg and Kumar, 2003; Xia, 2000; Xia *et al.*, 2003a).

3.2 Amino acid-based and codon-based substitution models

Amino acid-based models (Adachi and Hasegawa, 1996; Kishino *et al.*, 1990) are similar in form to those nucleotide-based models in the previous section, except that the discrete states of the Markov chain will be 20 instead of only 4. Because of the large size of the transition matrix, the transition probabilities are typically derived from empirical transition matrices (Dayhoff *et al.*, 1978; Jones *et al.*, 1992).

There are three inherent difficulties with amino acid-based models. First, protein-coding genes often differ much in substitution patterns, and one can never be sure if any of the empirical transition matrices is appropriate for the protein sequences one is studying. Second, note that an amino acid replacement is effected by a nonsynonymous codon replacement. Two codons can differ by 1, 2, or 3 sites, and an amino acid replacement involving two codons differing by one site is expected to be more likely than that involving two codons differing by 3 sites. However, at the amino acid level, there is no information on whether an amino acid replacement results in a single nucleotide replacement or a triple nucleotide replacement. Only a codon-based model can incorporate this information. Third, two similar amino acids are expected to, and do, replace each other more frequently than two different amino acids (Xia and Li, 1998). However, the similarity between amino acids is difficult to define. For example, polarity may be highly conserved at some sites but not at others. Two very different amino acids rarely replace each other in functionally important domains but can replace each other frequently at unimportant segment. Moreover, the likelihood of two amino acids replacing each other also depends on neighboring amino acids (Xia and Xie, 2002). For example, whether a stretch of amino acids will form a α-helix may depend on whether the stretch contains a high proportion of amino acids with high helix-forming propensity, and not necessarily on whether a particular site is occupied by a particular amino acid.

The codon-based substitution models (Goldman and Yang, 1994; Muse and Gaut, 1994) were proposed to overcome some of the difficulties in amino acid-based models. These models share the third difficulty above with the amino acid-based models, and have additional problems of their own. For example, one cannot get good estimate of codon frequencies because protein-coding genes are typically very short. An alternative is to use the F3x4 codon frequency model (Yang, 2002; Yang and Nielsen, 2000). However, codon usage is affected by many factors, including differential ribonucleotide and tRNA abundance as well as biased mutation (Xia, 1996, 1998b, 2005c). For example, the site-specific nucleotide frequencies are poor predictors of codon usage (Table 13-1) of protein-coding genes in *Escherichia coli* K12.

Table 13-1. Site-specific nucleotide frequencies and codon usage in two codon families. AA – amino acid; N_{cod} – number of codon; Results based on eight highly expressed genes (gapC, gapA, fbaB, ompC, fbaA, tufA, groS, groL) from the Escherichia coli K12 genome (GenBank Accession: NC_000913)

Nuc. Freq. by codon sites (CS)					Codon freq.		
Base	CS_1	CS_2	CS_3		Codon	AA	N_{cod}
A	0.273	0.32	0.18		AAG	Lys	24
C	0.189	0.24	0.326		AAA	Lys	149
G	0.409	0.16	0.219		CAG	Gln	73
U	0.129	0.28	0.275		CAA	Gln	7

A-ending codon are used frequently for coding lysine, but G-ending codon used frequently for coding glutamine (Table 13-1). The reason for this is simple. Six Lys-tRNA genes in *E. coli* K12 all have anticodons being UUU which can translate the AAA lysine codon better than the AAG lysine codon. For glutamine codons, there are two copies of Glu-tRNA genes (glnX and glnV) with a CUG anticodons matching the CAG codon and another two copies (glnW and glnU) with the UUG anticodon matching the CAA codon. However, the former is more abundant than the latter in the *E. coli* cell (Ikemura, 1992), which would favor the use of CAG against the CAA codon for glutamine. One should expect the F3x4 codon frequency model to perform poorly in such a situation which unfortunately is frequently encountered.

The complex array of factors contributing to codon usage bias is illustrated in Chapter 9. Readers should refer to that chapter in order to appreciate the difficulty in developing realistic codon-based models as well as the poor performance of several proposed codon-based models in molecular phylogenetics.

4. TREE-BUILDING METHODS

Three categories of tree-building methods are in common use: the distance-based, the maximum parsimony and the maximum likelihood methods. These methods have their respective advantages and disadvantages and I will provide mathematical details for the reader to understand their problems.

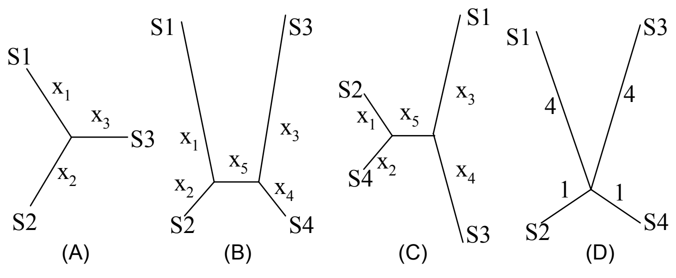
4.1 Distance-based methods

The distance-based methods build trees from a distance matrix, and are represented by UPGMA (Sneath, 1962), the neighbor-joining (NJ) method (Saitou and Nei, 1987), the Fitch-Margoliash (FM) method (Fitch and Margoliash, 1967) and the FastME method (Desper and Gascuel, 2002). The calculation of some genetic distances has already been covered in previous sections. Other genetic distances include the paralinear (Lake, 1994) and LogDet (Lockhart *et al.*, 1994) distances. A variety of genetic distances, together with tree-building algorithms such as UPGMA, NJ, FM and FastME (an excellent representative of distance-based methods based on the minimum criterion), are implemented in my program DAMBE (Xia, 2001; Xia and Xie, 2001b).

Other than the simplest UPGMA method, each tree-building method consists of two steps: (1) the evaluation of branch lengths for a given topology by either the least-squares (LS) method, the NJ method or the FM method, and (2) the selection of the best tree based on either the minimum evolution (ME) criterion or the least-squares or the weighted least-squares criterion referred to hereafter as the Fitch-Margoliash (FM) criterion. One should not confuse, e.g., the FM way of evaluating branch lengths with the FM criterion for choosing the best tree.

There are many ways of evaluating branch lengths for a given tree, and I will only present the LS method here. For the three-OTU (operational taxonomic unit) tree in Figure 13-2A, the branch lengths (x_i) can be solved uniquely by the following equations:

$$\begin{aligned} d_{12} &= x_1 + x_2 \\ d_{13} &= x_1 + x_3 \\ d_{23} &= x_2 + x_3 \ . \end{aligned} \tag{13.40}$$



	S2	S3	S4
S1	5	8	5
S2		5	2
S3			5
S1	4	5	4
S2		4	2
S3			4
S1	6	12	6
S2		6	2
S3			6

Figure 13-2. Topologies for illustrating the distance-based methods.

For the four-OTU tree in Figure 13-2B, we can write down the equations in the same way as in Eq. (13.40), but there will be six equations for five unknowns. The LS method finds the x_i values that minimize the sum of squared deviations (*SS*),

$$SS = \sum (d_{ij} - d'_{ij})^2 = [d_{12} - (x_1 + x_2)]^2 + ... + [d_{34} - (x_3 + x_4)]^2 \ . \tag{13.41}$$

By taking the partial derivatives with respect to x_i, setting the derivatives to zero and solving the resulting simultaneous equations, we get

$$\begin{aligned} x_1 &= d_{13}/4 + d_{12}/2 - d_{23}/4 + d_{14}/4 - d_{24}/4 \\ x_2 &= d_{12}/2 - d_{13}/4 + d_{23}/4 - d_{14}/4 + d_{24}/4 \\ x_3 &= d_{13}/4 + d_{23}/4 + d_{34}/2 - d_{14}/4 - d_{24}/4 \\ x_4 &= d_{14}/4 - d_{13}/4 - d_{23}/4 + d_{34}/2 + d_{24}/4 \\ x_5 &= - d_{12}/2 + d_{23}/4 - d_{34}/2 + d_{14}/4 + d_{24}/4 + d_{13}/4 \ . \end{aligned} \tag{13.42}$$

With four OTUs, there are three unrooted trees. There are two commonly used global criteria for choosing the best tree. The first is the ME criterion based on the tree length (*TL*) which is the summation of all x_i values. The tree with the smallest *TL* is chosen as the best tree. Note that TL can be computed directly from d_{ij} values without first evaluating x_i values.

In contrast, the FM criterion chooses the tree with the smallest *SS*

$$SS = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{\left(d_{ij} - d'_{ij}\right)^2}{d_{ij}^P} \tag{13.43}$$

where n is the number of OTUs and P often takes the value of 0 or 2.

Whether a distance-based method will recover the true tree depends critically on the accuracy of the distance estimates. We will briefly examine this problem with both the ME criterion and the FM criterion. Let TL_B and TL_C be the tree length for Trees B and C in Figure 13-2. Suppose that OTUs 1 and 3 have diverged from each other so much as to have experienced substitution saturation (Xia *et al.*, 2003b) to cause difficulty in estimating the true D_{13}. Let pD_{13} be the estimated D_{13}, where p measures the degree of underestimation ($p < 1$) or overestimation ($p > 1$). Designate D_{TL} as the difference in *TL* between the two trees,

$$D_{TL} = TL_B - TL_C = \frac{d_{12} + d_{34} - (pd_{13} + d_{24})}{4} \ . \tag{13.44}$$

According to the LS method of branch evaluation, Tree B is better than Tree C if $D_{TL} < 0$, and worse than Tree C if $D_{TL} > 0$. Simple distances such as the p-distance or JC69 distance tend to have $p < 1$ and consequently increase the chance of having $D_{TL} > 0$, i.e., favoring the incorrect Tree C. This is the long-branch attraction problem, first recognized in the maximum parsimony method (Felsenstein, 1978b). Genetic distances corrected with the gamma-distributed rates over sites (Golding, 1983; Jin and Nei, 1990; Nei and Gojobori, 1986; Tamura and Nei, 1993) tend to have $p > 1$ when there is in fact no rate heterogeneity over sites, and consequently would favor Tree B over Tree C, leading to long-branch repulsion (Waddell, 1995).

The long-branch attraction and repulsion problem is also present with the FM criterion. Let SS_B and SS_C be *SS* in Eq. (13.43) for Trees B and C, respectively. With P = 0 in Eq. (13.43) and letting $D_{SS} = SS_B - SS_C$, we have

$$\begin{aligned} 4D_{SS} &= (d_{13} + d_{24})^2 - (d_{12} + d_{34})^2 + 2(d_{14} + d_{23})[(d_{12} + d_{34}) - (d_{13} + d_{24})] \\ &= x^2 - y^2 + 2z(y - x) \end{aligned} \tag{13.45}$$

where $x = d_{13}+d_{24}$, $y = d_{12}+d_{34}$ and $z = d_{14}+d_{23}$.

We now focus on Tree D, for which y is expected to equal z. Now Eq. (13.45) is reduced to

$$4D_{SS} = (x - y)^2 \qquad (13.46)$$

If branch lengths are accurately estimated, then $x = y = 10$, and $D_{SS} = 0$, i.e., neither Tree B nor Tree C is favored. However, if d_{13} (i.e., the summation of the two long branches) is under- or overestimated, then $D_{SS} > 0$ favoring Tree C. This means that both under- and overestimation of the distance between divergence taxa will lead to long-branch attraction. This can be better illustrated with a numerical example with Tree D in Figure 13-2 which also displays three distance matrices. The first one is accurate, the second one has genetic distances more underestimated for more divergent taxa, and the third has genetic distances more overestimated for more divergent taxa (e.g., when gamma-distributed rates are assumed when the rate is in fact constant over sites). Note that Tree B and Tree C converge to Tree D when $x_5 = 0$. Table 13-2 shows the results by applying the ME and LS criterion in analyzing the three distance matrices.

When the distances are accurate, the application of both the ME criterion and the FM criterion recovers Tree D (the true tree) with $x_5 = 0$, $TL = 10$, and $SS = 0$. However, ME criterion favors Tree C when long branches are underestimated, and Tree B when long branches are overestimated. In contrast, the FM criterion would favor Tree C with both under- and overestimated distances (Table 13-2) when negative branches are allowed.

Table 13-2. Effect of under- and over-estimation of genetic distances.

	Correct		Under-estimation		Over-estimation	
	TreeB	TreeC	TreeB	TreeC	TreeB	TreeC
TL	10	10	7.75	7.5	12.5	13
SS	0	0	0.25	0	1	0
x_5	0	0	-0.25	0.5	0.5	-1

Distance-based methods, when used together with properly estimated genetic distances, are generally robust against various biases that handicap the maximum parsimony methods (illustrated in the next section). However, a recent paper has suggested a few topological biases associated with commonly used distance methods (Xia, 2006) which may be summarized briefly below.

With two alternative unlabelled topologies (tree shapes, designated as Topology A and Topology B, respectively) for six OTUs (Figure 13-3), we have 105 possible unrooted labeled topologies.

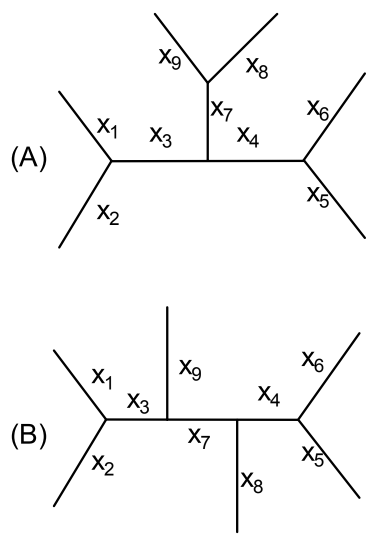


Figure 13-3. Two unlabelled topologies (tree shapes), with Topology A having three cherries and Topology B having two (a cherry is a pair of adjacent leaves descending from the most recent common ancestor).

There are 15 different ways of assigning the six OTUs to the leaves in Topology A and 90 different ways of assigning the six OTUs to Topology B. If we use randomly generated distance matrices, and if the distance-based methods are not biased in favor of one tree shape against the other, then the probability of getting Topology A and Topology B should be p = 15/105 and q = 90/105, respectively. However, the observed p is greater than 15/105 (and observed q smaller than 90/105), when either the neighbor-joining (Saitou and Nei, 1987), FastME (Desper and Gascuel, 2002) or Fitch-Margoliash method (Fitch and Margoliash, 1967) is used. This suggests that these distance methods may be biased in favor of topologies with more cherries (a cherry is a pair of adjacent leaves descending from the most recent common ancestor).

The bias may be explained by the way the x_i values are estimated by the least-squares method. Designate d_{ij} as the genetic distance between two OTUs i and j, and d_{ij}' the sum of branches linking OTUs i and j on the reconstructed tree. When there are only three OTUs, the three x_i values can be solved exactly and d_{ij} is always equal to d_{ij}'. With more than three OTUs, d_{ij} is equal to d_{ij}' only when OTUs i and j are in the same cherry, i.e., separated by only one internal node. A tree with more cherries with have more d_{ij} values equal to d_{ij}' and may consequently be favored.

It may not be obvious that d_{ij} = d_{ij}' when OTUs i and j are in the same cherry. So a bit of explanation is due. For the UPGMA method, whenever two OTUs i and j are clustered, the branching point is set at a distance equal to $d_{ij}/2$. So $d_{ij} = d_{ij}$'. For the NJ method, with two OTUs A and B separated by a single internal node X, the branch lengths between A and X and between B and X, designated b_{AX} and b_{BX}, respectively, are computed by the NJ method as

$$\begin{aligned} b_{AX} &= \frac{1}{2(N-1)}\left[(N-2)d_{AB} + R_A - R_B\right] \\ b_{AX} &= \frac{1}{2(N-1)}\left[(N-2)d_{AB} + R_B - R_A\right] \end{aligned} \qquad (13.47)$$

where N is the number of OTUs, $R_A = \Sigma d_{Ai}$, and RB = Σd_{Bi} (Saitou and Nei, 1987; Studier and Keppler, 1988). It is now obvious that d_{AB}' = ($b_{AX} + b_{BX}$) = d_{AB}. These branch lengths, i.e., b_{AX} and b_{BX} specified in Eq. (13.47), are also known to be the least-square estimates (Saitou and Nei, 1987).

For the Fitch-Margoliash method (Fitch and Margoliash, 1967) with two OTUs A and B separated by a single node X, the branch lengths are computed by first merging all the rest of OTUs into a single OTU, designated C, and then using the following equation with x, y, and z designating the branch lengths between A and X, B and X and C and X, respectively,

$$\begin{aligned} x &= (d_{AB} + d_{AC} - d_{BC})/2 \\ y &= (d_{AB} + d_{BC} - d_{AC})/2 \\ z &= (-d_{AB} + d_{AC} + d_{BC})/2 \end{aligned} \qquad (13.48)$$

Eq. (13.48) shows clearly that d_{AB}' = (x + y) = d_{AB}. For three OTUs, i.e., when C is not a composite OTU, these branch lengths are also least-square estimates.

When $d_{ij} = d_{ij}$', it does not contribute to SS in Eq. (13.41) or SS in Eq. (13.43). It is possible that Topology A in Figure 13-3 with three d_{ij} values equal to their corresponding d_{ij}' values, may be favored than Topology B in Figure 13-3 with only two d_{ij} values equal to their corresponding d_{ij}' values.

A related, but slightly different explanation for the possible bias in favor of topologies with more cherries is as follows. Because the least-squares method is a minimization method, the more often the observed d_{ij} values appear in equations for estimating x_i values, the more constraints are imposed on the minimization. If we expressed the nine x_i's as functions of d_{ij} values in the same form as Eq. (13.42), we will find the observed d_{ij} values appearing in the nine functions 93 times for Topology A but 95 times for Topology B. Thus, the minimization problem associated with Topology B is subject to more constraints than that with Topology A. Whether this can explain why Topology A is favored needs more study.

4.2 Maximum parsimony methods

The maximum parsimony method saw its first effective algorithm in 1971 (Fitch, 1971), and became immensely popular mainly due to the excellent software package PAUP (Swofford, 1993). The method remains the best entry point for computer science students to learn molecular phylogenetics and for biology students to develop a basic vocabulary of computation for communicating with programmers.

4.2.1 The Fitch algorithm

In contrast to the distance-based methods, maximum parsimony (MP) and maximum likelihood methods are character-based methods. The six aligned sequences in Figure 13-4 have nine sites, with sites 2, 4, 9 being monomorphic, and the rest of sites being polymorphic. A polymorphic site with at least two different states each represented by at least two OTUs is defined as an informative site. The MP method operates on informative sites only in its search for the best tree.

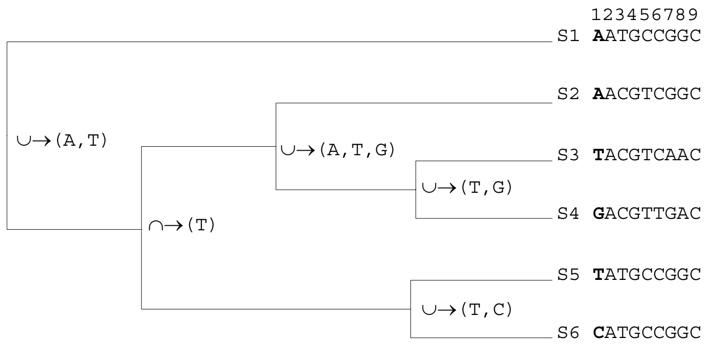


Figure 13-4. Computing the minimum number of changes for the first site of the six alignment sequences in phylogenetic reconstruction using the maximum parsimony method

Given a topology, we compute the minimum number of changes for each sequence site, with the computation of the first site illustrated in Figure 13-4. Each node is represented by a set of characters, with the terminal nodes (leaves) each represented by a set containing a single character. The method traverses through each internal node, starting from the node closest to the leaves. If two sets of the two daughter nodes have an empty intersection, then the node will be represented by the union of the two daughter sets, otherwise the node will be represented by the intersection. Once the operation reaches the root, then the number of union operations is the minimum number of changes needed to map the site to the tree.

Site 1 in Figure 13-4 requires four union operations (Figure 13-4), whereas sites 3, 5, and 8 each require only one union operation. Sites 6 and 7, which are polymorphic with two nucleotide states but not informative, will require one change for any topology. So the minimum number of changes, also referred to as the tree length, given the topology and the sequences in Figure 13-4, is nine. The same computation is done for other possible topologies and the tree with the smallest tree length is taken as the MP tree.

One may think that it is necessary to have a 4×L matrix (where L is sequence length) to store the possible ancestral states in the internal nodes for nucleotide sequences with four bases. In practice, only a vector of L elements is sufficient to store the states of the internal nodes by using ambiguous coding notation similar to the IUB (International Union of Biochemistry) code in Table 2-2 in Chapter 2. For example, according to the IUB code, (A,T) is coded as W, (A, T, G) as D, (T,G) as K and (T, C) as Y. If we compare the first nucleotide between S5 and S6 (T and C, respectively), we would need to look up the Table 2-2 of ambiguous codes to find letter Y. Looking up the ambiguous code table for every set of nucleotides in the internal node is time consuming.

To speed up the computation, one will almost always use bitwise operations. Here I illustrate the implementation in the DNAPARS program in the PHYLIP package (Felsenstein, 2002). First, for nucleotide i, where i = A, C, G, T, O (for **o**ther such as the gap "-"), we define I_i (i.e., I_A, I_C, I_G, I_T and I_O) as 0, 1, 2, 3, and 4, respectively. We also note that the binary notation of the decimal value of 1 is "00000001" which, when left-shifted by I_A, I_C, I_G, I_T or I_O bits, yields decimal values 1, 2, 4, 8 and 16, respectively (Table 13-3).

Table 13-3. Decimal values obtained by left-shifting (<<) the binary 00000001 by I_i.

Nuc	I_i	$1<<I_i$	Decimal
A	0	00000001	1
C	1	00000010	2
G	2	00000100	4
T	3	00001000	8
O	4	00010000	16

We can now use bitwise operations to obtain the union and intersection. In almost all programming languages, there is an operation called bitwise OR that will combine corresponding bits of two binary numbers in such a way that the result is 1 if either of the operand bits is 1, and is 0 when both operand bits are 0. The bitwise OR is typically designated by | in C-related languages and simply OR in VB (Visual Basic)-related languages. Thus, the union of A (represented in binary as 00000001) and G (represented in binary as 00000100) is 00000001 | 00000100 = 00000101, which equals a decimal value of 5. We consequently assign the value of 5 to the letter R (for either A or G according to the IUB coding).

Similarly, the union of C and T is 00000010 | 00001000 = 00001010, which equals a decimal value of 10. We consequently assign the value of 10 to the letter Y (for either Y or T according to IUB coding). The union of R and Y is N (any of the four nucleotides), obtained by 00000101 | 00001010 = 00001111 which equals the decimal value of 15. The letter N consequently gets the value of 15.

In VB-related languages, we have R = (1 OR 4) = 5 for the union of A and G, Y = (2 OR 8) = 10 for the union of C and T, N = (5 OR 10) =15, and so on. To help you check the output of the bitwise operations, I have listed the nucleotides, their respective ascii keycodes, assigned values with binary notation and the meanings of ambiguous codes in Table 13-4.

Table 13-4. Nucleotides (Nuc), their ascii code, assigned decimal (binary) values according to bitwise operations and their meanings.

Nuc	Ascii	Value	Meaning
A	65	1 (00000001)	A
C	67	2 (00000010)	C
M	77	3 (00000011)	A or C
G	71	4 (00000100)	G
R	82	5 (00000101)	A or G
S	83	6 (00000110)	C or G
V	86	7 (00000111)	A or C or G
T	84	8 (00001000)	T
W	87	9 (00001001)	A or T
Y	89	10 (00001010)	C or T
H	72	11 (00001011)	A or C or T
K	75	12 (00001100)	G or T
D	68	13 (00001101)	A or G or T
B	66	14 (00001110)	C or G or T
N	78	15 (00001111)	G or A or T or C
O	79	16 (00010000)	- or whatever else

Programming languages also feature a bitwise AND operator to help us get intersection. It is represented as & in C-related languages and simply AND in VB-related languages. It combines corresponding bits of two binary numbers in such a way that the result is 1 if both operand bits are 1, and is 0 otherwise. Thus, the intersection of A and R is (A & R) = 1 in C-related languages, (A AND R) = 1 in VB-related languages (Recall that 1 is the number assigned to A (Table 13-4), so the operation simply means that the intersection of A and R is just A.

The brief introduction of binary operations above suggests a clever trick in speeding up the evaluation of a particular tree topology. We define

```
S = "ACMGRSVTWYHKDBNO???????????????"
```

so that A = 1, C = 2, M = 3, G = 4, …, N = 15 (which stands for any nucleotide), O = 16, and "?" = 31, then the union and intersection of the two child nodes can be obtained easily by the bitwise OR and bitwise AND operations. For example, if the nucleotide at the left child node is A = 1 and the right child node is K = 12 (i.e., T or G), then their union at the parental node is 1 | 12 = 13 (which is D standing for A, G or T). Similarly, if the nucleotide at the left child node is (T,C) = Y = 10 and the right child node is (A, G, T) = D = 13, then their intersection at the parental node is 10 & 13 = 8 (which is the number assigned to T). Bitwise operations are very fast, even for VB-related languages.

The evaluation of each tree can also be sped up by first collapsing the sites into site patterns. When sequences are long and when the number of OTUs few, most sites will have the same site pattern. For example, if all nine columns of data in Figure 13-4 are the same as the first column, then we only need to evaluate the first column and multiple the number of changes by nine to get the tree length. However, when the number of sequences increases, this approach becomes less and less effective the probability of different sites sharing the same site pattern decreases with the number of sequences.

In summary, the algorithm for the MP method consists of two components, one being a tree generation function to generate alternative topologies and the other being the tree evaluation function to output the minimum number of changes for each alternative tree topology. The topology with the smallest number of changes is the MP tree.

4.2.2 The uphill search and branch-and-bound search algorithms

The number of possible topologies (Felsenstein, 1978a) increases very quickly with the increase in the number of OTUs (n), with the number of rooted and unrooted topologies, designated as N_R and N_U, respectively, being

$$\begin{aligned} N_R &= \frac{(2n-3)!}{2^{n-2}(n-2)!} \\ N_U &= \frac{(2n-5)!}{2^{n-3}(n-3)!} \end{aligned} \tag{13.49}$$

Searching through all possible topologies is called exhaustive search. It is often computationally impossible to search all possible trees with a large n. Two faster alternatives are commonly used. The first is the uphill search which does not guarantee that the resulting tree is the most parsimonious. The second is the branch-and-bound approach which does guarantee the finding of the most parsimonious tree.

The uphill search algorithm is illustrated in Figure 13-5 with rooted topologies. We first take three OTUs and evaluate all three possible alternative topologies. If T1 (Figure 13-5) is the shortest of the three, then we ignore the other two topologies (i.e., T2 and T3 in Figure 13-5) and all other 4-OTU topologies derived from them. Now we add the fourth OTU at all possible positions of T1 and generate the five possible topologies. Evaluating these five topologies generates the 4-OTU "MP tree".

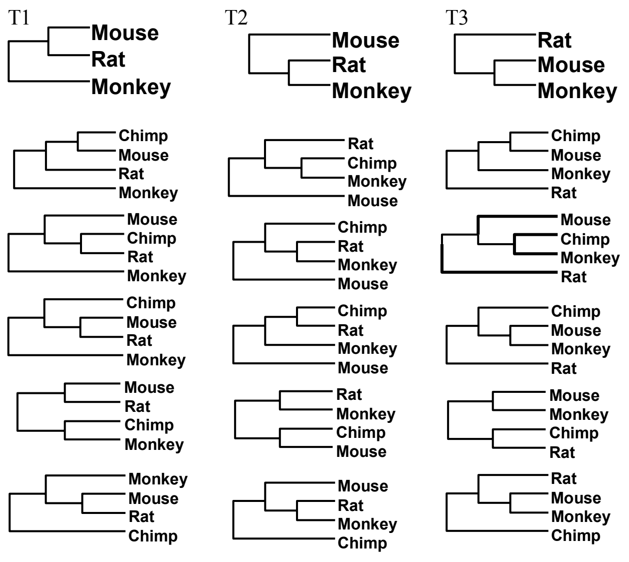


Figure 13-5. Illustration of the uphill and branch-and-bound searching algorithm. T1, T2 and T3 designate the three 3-OTU topologies. Adding the fourth OTU at all possible positions in T1 generates the five 4-OTU topologies under it. The same for T2 and T3 and their respective 4-OTU topologies.

Compared to the exhaustive search evaluating all 15 possible 4-OTU topologies, the uphill search is obviously much faster. Computation simulations have shown that the uphill search generates optimal or near-optimal trees.

The branch-and-bound algorithm for the MP method (Hendy and Penny., 1982) starts by generating an initial tree with fast algorithms such as the uphill search. Designate the length of the initial tree (i.e., the number of changes required to account for the sequence variation) as L, which is the upper bound of the tree length for the true MP tree. One may also use the neighbor-joining method (Saitou and Nei, 1987) to generate a topology and then evaluate the topology to obtain L. The resulting topology also allows us to know which OTUs tend to have long branches.

The next step is to rank OTUs according to their branch lengths and take the three OTUs with the longest branch lengths to build the 3-OTU topologies. More divergent OTUs are added to the tree earlier. Any subtree with a tree length > L is eliminated together with all topologies derived from such a subtree because adding more OTUs will only lengthen the tree. This is why we use more divergent OTUs first because this increases the chance of having subtrees with a tree length > L so that such subtrees get eliminated early. Take topologies in Figure 13-5 for example. If L = 10 and the 3-OTU topologies T2 and T3 already have tree lengths greater than L, then we do not need to consider the 4-OTU topologies derived from them because such 4-OTU topologies cannot be the MP tree. This process continues until the true MP tree is found.

The efficiency of the branch-and-bound algorithm depends much on the initial topology. If the initial topology is good, then many non-MP alternatives gets eliminated early. If it is bad, then the algorithm will be nearly as slow as the exhaustive search.

4.2.3 The long-branch attraction problem

The MP method is known to be inconsistent (Felsenstein, 1978b; Takezaki and Nei, 1994) and I will provide a simple demonstration here by using trees in Figure 13-6. With four species, we have three possible unrooted topologies, designated T_k (k = 1, 2, 3), with T_1 being the correct topology.

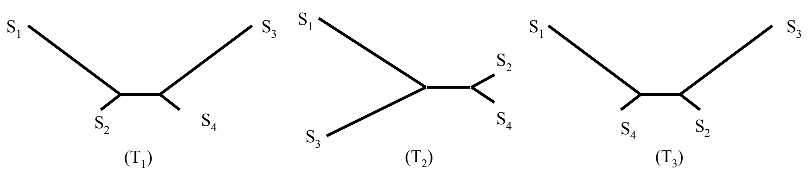


Figure 13-6. The long-branch attraction problem in the maximum parsimony methods.

Let X_{ij} be nucleotide at site j for species S_i, and L be the sequence length. For simplicity, assume that nucleotide frequencies are all equal to 0.25. Suppose that the lineages leading to S_1 and S_3 have experienced full substitution saturation, so that

$$\Pr(X_{1j} = X_{ij,i\neq 1}) = \Pr(X_{3j} = X_{ij,i\neq 3}) = 0.25 \tag{13.50}$$

where *Pr* stands for probability. The lineages leading to X_2 and X_4 have not experienced substitution saturation and have

$$\Pr(X_{2j} = X_{4j}) = P \tag{13.51}$$

where $P > 0.25$. For simplicity, let us set $P = 0.8$, and $L = 1000$.

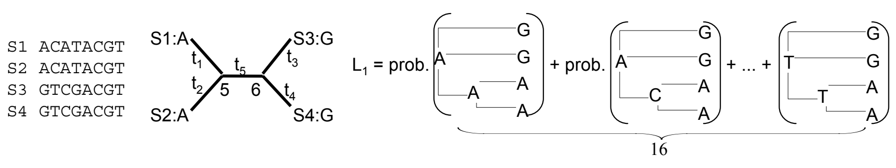
We now consider the expected number of informative sites, designated by n_k (k = 1, 2, 3), favoring T_k. By definition, site j is informative and favoring T_1 if it meets the following three conditions: $X_{1j} = X_{2j}$, $X_{3j} = X_{4j}$, $X_{1j} \neq X_{3j}$. Similarly, site j favors T_2 if $X_{1j} = X_{3j}$, $X_{2j} = X_{4j}$, $X_{1j} \neq X_{2j}$. Thus, the expected numbers of informative sites favoring T_1, T_2 and T_3, respectively, are

$$\begin{aligned} E(n_1) &= \Pr(X_{1j} = X_{2j}, X_{3j} = X_{4j}, X_{1j} \neq X_{3j})L \\ &= 0.25 \times 0.25 \times 0.75 \times 1000 \approx 47 \\ E(n_2) &= \Pr(X_{1j} = X_{3j}, X_{2j} = X_{4j}, X_{1j} \neq X_{2j})L \\ &= 0.25 \times 0.8 \times 0.75 \times 1000 = 150 \\ E(n_3) &= E(n_1) \approx 47\ . \end{aligned} \tag{13.52}$$

The equations mean that, in spite of T_1 being the true topology, we should have, on average, only about 47 informative sites favoring T_1 and T_3, but 150 sites supporting the wrong tree T_2. This is one of the several causes for the familiar problem of long-branch attraction (Hendy and Penny, 1989) or short-branch attraction (Nei, 1996). Because it is the two short branches that contribute a large number of informative sites supporting the wrong tree, "short-branch attraction" seems a more appropriate term for the problem than "long-branch attraction".

4.3 Maximum likelihood methods

The maximum likelihood (ML) method is introduced into molecular phylogenetics by Joe Felsenstein (1981) and popularized by his DNAML program in his PHYLIP package (Felsenstein, 2002). It is based on explicit substitution models. Many different types of computer simulation have demonstrated the superiority of the ML method in recovering the true tree. I now use the four aligned sequences in Figure 13-7 to illustrate numerically the computation involved in the ML method based on the JC69 model. With four sequences, we have three possible unrooted topologies of which one is shown in Figure 13-7.



```
S1 ACATACGT
S2 ACATACGT
S3 GTCGACGT
S4 GTCGACGT
```

Figure 13-7. Likelihood calculation for the first site of the four aligned sequences.

We do not know the state of the two internal nodes, labeled as nodes 5 and 6, respectively, in Figure 13-7. So we need to consider all four possibilities (i.e., A, C, G, and T) for each node, with 16 possible combinations (Figure 13-7). Note that the number of possible combinations increases rapidly as 4^N, where N is the number of internal nodes. This is one of main reasons why likelihood methods in phylogenetics are slow.

The sequences have 8 sites, with the first four sites sharing one site pattern and the last four sites sharing another site pattern. So we need only two site-specific likelihood functions. You may recall that the JC69 model assumes that all nucleotides substitute each other with equal probabilities and that nucleotide frequencies are equal. This is why we can treat the first four sites as having the same site pattern.

The likelihood function of the first site, given the topology in Figure 13-7, is the summation of the 16 probabilities corresponding to the 16 nucleotide combinations of the two internal nodes with unknown nucleotides (Figure 13-7). Thus, the likelihood of the first site is,

$$\begin{aligned} L_1 = {} & \pi_A P_{AA.t_1} P_{AA.t_2} P_{AA.t_5} P_{AG.t_3} P_{AG.t_4} \\ & + \pi_C P_{CA.t_1} P_{CA.t_2} P_{CA.t_5} P_{AG.t_3} P_{AG.t_4} \\ & + \ldots \\ & + \pi_T P_{TA.t_1} P_{TA.t_2} P_{TT.t_5} P_{TG.t_3} P_{TG.t_4} \end{aligned} \quad (13.53)$$

where $P_{ij.t}$ for the JC69 model has already been given in Eq. (13.6) except that "8αt" should be replaced by "4αt" because here we are not dealing with branch lengths connecting two extant OTUs. Note that $L_2 = L_3 = L_4 = L_1$. We can write L_5 ($= L_6 = L_7 = L_8$) in a similar fashion.

The sequences in Figure 13-7 allow us to simplify Eq. (13.53) greatly. Note that S1 = S2 and S3 = S4 (Figure 13-7) so that αt_1, αt_2, αt_3, and αt_4 are all zero. Now we have

$$\begin{aligned} L_1 &= 0.0625 - 0.0625 e^{-4\alpha t_5} \\ L_5 &= 0.0625 + 0.1875 e^{-4\alpha t_5} \ . \end{aligned} \quad (13.54)$$

With the assumption that all sites evolve independently, the likelihood function for all eight sites is simply

$$\begin{aligned} L &= L_1^4 L_5^4 \\ \ln L &= 4\ln(L_1) + 4\ln(L_5) \\ &= 4\ln(0.0625 - 0.0625 e^{-4\alpha t_5}) + 4\ln(0.0625 + 0.1875 e^{-4\alpha t_5}) \ . \end{aligned} \quad (13.55)$$

The αt_5 value that maximizes ln*L* is 0.27465, which leads to ln*L* = -21.02998. The branch length between nodes 5 and 6 is $3\alpha t_5$ = 0.82396. We can do the same calculation for the other two possible topologies, and then choose the tree with the largest lnL value as the ML tree. In this particular example, the tree in Figure 13-7 is the ML tree because it has the lnL value greater than that of the other two trees. One may also find that the ML tree, including its estimated branch lengths, is identical to the tree from a distance-based method such as the neighbor-joining (Saitou and Nei, 1987), the FastME (Desper and Gascuel, 2002) or the Fitch-Margoliash method (Fitch and Margoliash, 1967) as long as the JC69 distance is used.

One can cite many advantages of the maximum likelihood over the maximum parsimony method, and one of the frequently cited advantages is that the former uses more information than the latter. The maximum parsimony method uses only the informative sites for searching the MP tree, whereas maximum likelihood methods, depending on the inherent substitution model adopted, can use information on all sites, including monomorphic sites.

There are two major criticisms on the ML method in phylogenetics. The first is that the application of the likelihood in phylogenetics is not really a ML method in its conventional sense because the topology is not in the likelihood function (Nei, 1987; Nei and Kumar, 2000). To see this point, we can illustrate the conventional ML method with a simple example.

Suppose we wish to estimate the proportion of males (*p*) of a fish population in a large lake. A random sample of *N* fish contains *M* males. With the binomial distribution, the likelihood function is

$$L = \frac{N!}{M!(N-M)!} p^M (1-p)^{N-M} \ . \qquad (13.56)$$

The maximum likelihood method finds the value of *p* that maximizes the likelihood value. This maximization process is simplified by maximizing the natural logarithm of L instead:

$$\begin{aligned} &\ln L = A + M\ln(p) + (N-M)\ln(1-p) \\ &\frac{\partial \ln L}{\partial p} = \frac{M}{p} - \frac{N-M}{1-p} = 0 \qquad (13.57) \\ &p = \frac{M}{N} \ . \end{aligned}$$

The likelihood estimate of the variance of *p* is the negative reciprocal of the second derivative,

$$Var(p) = -\frac{1}{\frac{\partial^2 \ln(L)}{\partial p^2}} = -\frac{1}{-\frac{M}{p^2} - \frac{N-M}{(1-p)^2}} = \frac{p(1-p)}{N} \ . \qquad (13.58)$$

Note that, in contrast to the likelihood in Eq. (13.57) which is a function of *p* (the parameter to be estimated), the likelihood in Eq. (13.55) does not have the topology as a parameter. Without the convenient "$\partial \ln L/\partial \theta = 0$" formulation, we have to do either exhaustive or branch-and-bound search in order to find the topology that maximizes the likelihood. In practice, exhaustive or branch-and-bound search is rarely done, which implies that few of the published ML trees are authentic ML trees. Thus, Nei's criticism highlights more of a practical difficulty than a theoretical one because the likelihood principle does not require the parameter to be continuous and differentiable (Chang, 1996). The criticism can also be applied to other phylogenetic methods. However, other methods are generally faster and can search the tree space more thoroughly than the ML method. Therefore, while it is not particularly controversial to claim that an authentic ML tree is generally better than a tree satisfying the MP, ME or FM criterion, it is not unreasonable for one to expect the latter to be as good as or better than a "ML" tree that is obtained from searching a small subset of all possible topologies. This is particularly pertinent with reconstructing very large phylogenies (Tamura *et al.*, 2004).

The second criticism is on the assumptions shared by nearly all the substitution models currently implemented in the likelihood framework: (1) the substitutions occur independently in different lineages, (2) substitutions occur independently among sites, and (3) the process of substitution is described by a time-homogeneous (stationary) Markov process. The likelihood depends on the assumptions of the substitution model, and we generally cannot be sure if the model we use is appropriate.

Among the three assumptions mentioned above, the first assumption is false in taxa with a history of horizontal gene transfer which is rampant in bacterial species (Brown, 2003; Eisen, 2000; Koonin, 2003; Kurland *et al.*, 2003; Medigue *et al.*, 1991; Philippe and Douady, 2003) or with gene conversion which is ubiquitous (Aylon and Kupiec, 2004; Drouin, 2002a, 2002b; Drouin and de Sa, 1995; Drouin *et al.*, 1999; Hickey *et al.*, 1991).

The problem of the second assumption can be illustrated with the following example involving the GAT and GGT codons. Both codons end with a T. Whether a T→A substitution would occur depends much on whether the second position is an A or a G. The T→A substitution is rare when the second codon position is A because a T→A mutation in the GAT codon is nonsynonymous, but relatively frequent when the second codon position is G because such a T→A mutation in a GGT codon is synonymous. So nucleotide substitutions do not occur independently among sites (Xia, 1998a). This is one of the reasons for using codon-based models but these models have their own problems as mentioned before.

The third assumption is also problematic. Suppose we wish to reconstruct a tree from a group of orthologous sequences from both invertebrate and vertebrate species. There is little DNA methylation in invertebrate genomes, but heavy DNA methylation in some vertebrate genomes. DNA methylation greatly enhanced the C→T transition and consequently the G→A transition on the opposite strand (Xia, 2003). The net result is a much elevated transition/transversion bias and increased AT% in the lineages with DNA methylation, violating the third assumption.

The transfer of genes from a mitochondrial genome into a nuclear genome serves as another illustration of this third problem. The mutation spectrum and selection regime may differ substantially between the mitochondria and the nucleus, leading to nonstationarity. The gene transfer between mitochondrial genome and nuclear genome is an ongoing process in plants (Bonen, 2006; Bonen and Calixte, 2006).

More complicated models have been proposed in response to our increased knowledge of the substitution process. However, such parameter-rich models create two problems. First, the dependence of the likelihood value on tree topology decreases as the number of parameters increases because tree topology is just one of these parameters. So a better-fit model does not imply a more efficient recovery of the true tree. Second, parameter-rich models require more data for reliable parameter estimation. The dilemma is that increasing the sequence length also increases the heterogeneity of substitution processes (Xia, 1998a) including heterotachy (Kolaczkowski and Thornton, 2004) operating on different sequence segments and consequently increase the number of parameters to be estimated. Such heterogeneity over sites implies that the consistency of the ML method (Chang, 1996; Felsenstein, 1988) is not of much value because we cannot get long sequences for a fixed and small number of parameters. Take for example the estimation of the proportion of male fish in the lake. If we get only six male fish in a sample with no female, then the likelihood estimation of p is 1 which is worse than our wildest guess without any data.

4.4 Bayesian inference

The Bayesian approach has only recently been used extensively in phylogenetic inference (Aris-Brosou, 2003; Aris-Brosou and Yang, 2003; Huelsenbeck *et al.*, 2001) as well as in phylogeny-based detection of adaptive evolution (Aris-Brosou, 2005). In previous chapters, I have already illustrated Bayesian approach involving discrete variables. Here I illustrate (1) the basic principle of the Bayesian approach involving a continuous variable, and (2) three computational approaches to simplify the evaluation of posterior probabilities, i.e., the conjugate prior distributions, the discrete approximation the Markov chain Monte Carlo (MCMC) method.

4.4.1 Bayes theorem for a continuous variable

We will use the same example of estimating the proportion (p) of male fish in a fish population in a large lake. For a continuous variable such as p, the Bayes' theorem is

$$f(p \mid y) = \frac{f(y \mid p) f(p)}{\int f(y \mid p) f(p) dp} \tag{13.59}$$

where *p* is the parameter of interest, *y* is the observed sample data, *f*(*p*) is the prior probability density function for incorporating our prior knowledge on *p*, *f*(*y*|*p*) is the likelihood, and *f*(*p*|*y*) is the posterior probability. The numerator and the denominator are often referred to as the joint and marginal probabilities, respectively.

Suppose that we have taken a sample of six fish, all being males. Let N be the number of fish in the sample and M be the number of males in the sample. So we have N = 6 and M = 6. How should we use the Bayesian approach to estimate p (the proportion of males)?

We have three tasks to accomplish in order to obtain f(p|y). The first is to formulate f(p), our prior probability density function (referred hereafter as PPDF), the second is to formulate the likelihood, f(y|p), and the third is the most tricky, i.e., to get the integration in the denominator of Eq. (13.59).

The first task, i.e., formulate f(p), should have been done before taking the sample. According our conventional wisdom in vertebrate biology, the two sexes should be roughly equal in number (especially in a large lake where the fish population is most likely outbreeding). So the probability of p = 0.5 should be the largest and decrease as p becomes more extreme towards 0 or 1. Such a conventional wisdom can be described by the versatile beta distribution which is a two-parameter density function defined over the closed interval $0 \le p \le 1$ and used often as a model for proportions:

$$f(p) = \frac{(N'-1)!}{(M'-1)!(N'-M'-1)!} p^{M'-1} (1-p)^{N'-M'-1} \tag{13.60}$$

where N’ = 6 and M’ = 3 to reflect prior belief that p = 0.5 is the most likely. The primes in Eq. (13.60) in N’ and M’ indicate that they are for PPDF and different from N and M from the sample.

The PPDF expressed as the beta distribution with N’ = 6 and M’ = 3 is shown in Figure 13-8. If you think that the bell-shaped curve of PPDF reflects your prior wisdom, then we are in business and can continue with our computation. If you want to get another density function to replace f(p) in Eq. (13.60), please do so and we will continue just the same.

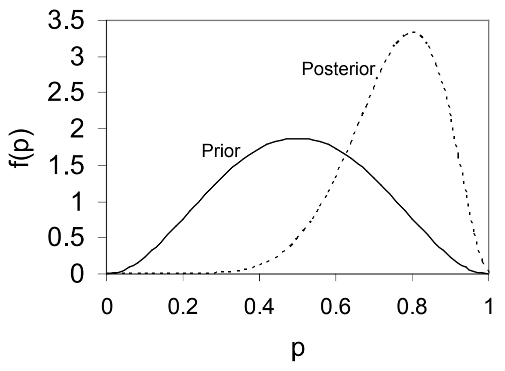


Figure 13-8. Comparison between prior and posterior probabilities.

The second task in evaluating f(p|y) is to formulate the likelihood function f(y|p), which is easy given the binomial distribution:

$$f(y\mid p)=\frac{N!}{M!(N-M)!}p^M(1-p)^{N-M} \tag{13.61}$$

If you do not like symbols, just substitute N' = 6, and M' = 3 into Eq. (13.60) and N = 6 and M = 6 into Eq. (13.61). Now the numerator of Eq. (13.59), designated as A, becomes

$$A=f(p)f(y\mid p)=30p^8(1-p)^2 \tag{13.62}$$

Finally we come to the more difficult third task, dealing with the denominator of Eq. (13.59). Fortunately for us, I have chosen perhaps the simplest possible problem for this illustration. The denominator, designated by B, is simply

$$B=\int_0^1 Adp=30p^8(1-p)^2dp=\frac{2}{33} \tag{13.63}$$

where A is given in Eq. (13.62). Now Eq. (13.59) is reduced to

$$f(p\mid y)=\frac{A}{B}=\frac{30p^8(1-p)^2}{2/33}=495p^8(1-p)^2 \tag{13.64}$$

The result is shown in Figure 13-8 in comparison with the prior probability. f(p|y) in Eq. (13.64) is a properly normalized probability density function as you can verify that

$$\int_0^1 495p^8(1-p)^2dp = 1 \tag{13.65}$$

The peak of f(p|y) is at p = 0.8, i.e., our prior expectation of p = 0.5 has been revised by the actual sample to p = 0.8. You may verify this by taking the derivative of f(p|y) in Eq. (13.64) with respect to p, setting the derivative to 0 and solving the resulting equation for p, which will yield p = 0.8.

You may have already noted that, for estimating a Bayesian p, we do not need the integral in the denominator of Eq. (13.59). You may verify this by taking the derivative of the numerator A in Eq. (13.62) with respect to p, setting the derivative to 0 and solving the resulting equation for p, which will also yield p = 0.8. It is only when we need to obtain f(p|y), the posterior probability density function, that we need the integral in denominator to normalize the numerator into a proper probability density function.

One misunderstanding from students is the following. The assumption of equal males and females inherent in the prior can be readily rejected by the sampling data of six males because the probability of the assumption being true is $0.5^6 = 0.015625$. So why should we use a very unlikely prior? To this question, a Bayesian can readily answer by pointing out that the prior should be properly assessed before the sampling takes place. However, this does not mean that accepting a prior probability density function is not controversial. As has been pointed out already (Felsenstein, 2004), many problems exist in choosing proper prior probabilities, and a Bayesian is characterized by being willing to accept controversial priors. Given the fact that posterior probabilities depend on prior probabilities, a Bayesian enjoys a great deal of flexibility in generating “desirable” results. Indeed, one may claim that no branch of statistics is more closely related to lies and damned lies than Bayesian statistics.

In our simple example, one may note that if the population of fish is indeed made of all males, e.g., when there is a high concentration of androgen masculinizing all individuals to males (Baron *et al.*, 2004), then the likelihood estimate of p = 1 is correct and the Bayesian estimate of p = 0.8 is wrong, and the wrong estimate may lead to our failure to identify an environmental crisis.

Aside from the controversy in setting prior probabilities, there are cases where priors can be assessed properly and there is no denial that Bayesian inferences have made significant contributions in virtually any branch of natural and social sciences where decision making is involved. For this reason, it is important to learn a few tricks that ease the computation burden of Bayesian inference.

4.4.2 Alternative computational approaches in Bayesian inference

One should know that, in practice, Eq. (13.59) is rarely used for computing the posterior probabilities because the integration in the denominator is difficult unless $f(\theta)$ and $f(y|\theta)$ are very simple (which luckily is true in our case). There are three alternative computational approaches. The first is to use the conjugate prior distributions (Raiffa and Schlaifer, 1961), the second is to use the discrete approximation, and the third is the MCMC (Markov chain Monte Carlo) approach (Hastings, 1970; Metropolis *et al.*, 1953) of which a special case, called Gibbs sampler, has already been presented in Chapter 7. I will briefly illustrate these approaches here using the same example.

A conjugate prior distribution is one that, after the mathematical operation specified in Eq. (13.59), will result in a posterior distribution, f(p|y), belonging to the same family of distributions as the prior. If you do not have a statistical handbook, Wikipedia lists many conjugate distributions under “Conjugate prior”. For our example involving a stationary and independent Bernoulli process in sampling fish, the conjugate prior distribution is the beta distribution already specified in Eq. (13.60), with N’ = 6 and M’ = 3 reflecting our prior knowledge that p is most likely 0.5. The density function of the prior is already shown in Figure 13-8.

Now we compute the posterior probability. It can be proven that, if the prior distribution of *p* is a beta distribution, then the posterior distribution will also be a beta distribution with the two parameters computed according to Eq. (13.66) below. In our actual sample with six males and 0 female (N = 6 and M = 6),

$$\begin{aligned} M'' &= M' + M = 3 + 6 = 9 \\ N'' &= N' + N = 6 + 6 = 12 \end{aligned} \qquad (13.66)$$

Now the posterior probability can be calculated by using Eq. (13.60) by substituting N’ and M’ in Eq. (13.60) with N" and M" in Eq. (13.66). The resulting f(p|y) is exactly the same as is specified in Eq. (13.64), i.e., you have derived f(p|y) without going through the hazardous step of integration. In particular, if you subsequently decided to take another sample of fish and get 4 males out of 7, all what you need to do is to use N’ = 12, M’ = 9, N = 7, and M = 4, and recalculate M’’ and N’’ using Eq. (13.66). So you can obtain the new posterior probability in no time.

The second alternative to the integration specified in the denominator of Eq. (13.59) is by the discrete approximation, which is illustrated in Table 13-5. Although variable p is continuous between 0 and 1, we have discretized it into 20 intervals, with $p_i = 0, 0.05, 0.1, \dots, 1$, and computed $f(p_i)$ according to Eq. (13.60) and $f(y|p_i)$ according to Eq. (13.61). The integral in the denominator of Eq. (13.59) is then approximated by

$$\int f(y \mid p) f(p) dp \approx \frac{\sum_{i=1}^{21} f(y \mid p_i) f(p_i)}{\sum_{i=1}^{21} f(p_i)} = \frac{1.21187329}{19.999875} = 0.0606 \qquad (13.67)$$

which is equal to the value of 2/33 in Eq. (13.63). Thus, we obtained the integral by simply taking a weighted arithmetic mean.

Table 13-5. Approximate the integral in Eq. (13.59) by discretization.

p_i	$f(p_i)$	$f(y\|p_i)$	$f(y\|p_i)*f(p_i)$
0	0	0	0
0.05	0.067688	0.000000	0.000000
0.1	0.243000	0.000001	0.000000
0.15	0.487688	0.000011	0.000006
0.2	0.768000	0.000064	0.000049
0.25	1.054688	0.000244	0.000257
0.3	1.323000	0.000729	0.000964
0.35	1.552688	0.001838	0.002854
0.4	1.728000	0.004096	0.007078
0.45	1.837688	0.008304	0.015260
0.5	1.875000	0.015625	0.029297
0.55	1.837688	0.027681	0.050868
0.6	1.728000	0.046656	0.080622
0.65	1.552688	0.075419	0.117102
0.7	1.323000	0.117649	0.155650
0.75	1.054688	0.177979	0.187712
0.8	0.768000	0.262144	0.201327
0.85	0.487688	0.377150	0.183931
0.9	0.243000	0.531441	0.129140
0.95	0.067688	0.735092	0.049757
1	0	1	0
Sum	19.999875		1.21187329

Discretizing variable p into finer intervals will have more accurate approximation. I will leave it as an exercise for you to discretize variable p into intervals of 0.01. This should be easy to do if you are familiar with any spreadsheet programs such as Microsoft EXCEL. Any practical computation would have involved smaller intervals. Choosing interval of 0.001 would imply the discretization of p into 1000 intervals.

For Bayesian inference involving a single variable, the discrete approximation works very well. However, the approach becomes clumsy with multiple variables. For example, with two variables of interval 0.001, we would need a 1000 by 1000 grid. Imagine the scenario of having a vector of 10 variables!

The third alternative is by Monte Carlo integration (MC integration), which brings us one step closer to MCMC algorithms. In our example, the denominator in Eq. (13.59) is approximated by MC integration as

$$\int f(y \mid p) f(p) dp = \frac{1}{n} \sum_{i=1}^{n} f(y \mid p_i) \tag{13.68}$$

where p_i are drawn randomly from the density f(p). This is in fact quite similar to the discrete approximation except that p_i is drawn from f(p) in MC integration in contrast to discretizing p into equal intervals in discrete approximation. Note that Eq. (13.68) and Eq. (13.66) are really the same except that the former is a plain arithmetic mean of f(y|p_i) and the latter is a weighted arithmetic mean of f(y|p_i).

For numerical illustration of MC integration, one would need a random number generator from the beta distribution. Readers well versed in computer languages such as Fortran, C or Visual Basic should be able to find such a random number generator in the web, or write one for themselves, or just request one from me. However, every university appears to have the software Maple installed. So one may just use the following commands to generate, say, 1000 random numbers from the beta distribution with N’ = 6 and M’ = 3:

```
with(stats):
stats[random, beta[3,3]](1000);
```

The resulting random numbers and associated f(p), f(y|p) and f(p)*f(y|p), computed according to Eqs. (13.60), (13.61) and (13.62), respectively, are partially shown in Table 13-6. The MC integral according to Eq. (13.68) is simply the mean of f(y|p), which in our case is 0.0604, slightly smaller than the exact integration in Eq. (13.63). The column headed by “f(p|y)” in Table 13-6 lists the resulting posterior probabilities, which are very close to the exact posterior probabilities (last column in Table 13-6) computed according to Eq. (13.64).

Table 13-6. Numerical illustration of Monte Carlo integration. The first column is the random number from the beta distribution. The MC integral is the arithmetic mean of the column headed by f(y|p) according to Eq. (13.68). The last column is the exact posterior probability from Eq. (13.64). The second last column is the posterior probabilities obtained by dividing f(p)*L by the MC integral.

p	f(p)	f(y|p)	f(p)*L	f(p|y)	$495p^8(1-p)^2$
0.797392458	0.783026967	0.257058956	0.201284095	3.33251813	3.321187569
0.365079937	1.611889587	0.002367706	0.003816481	0.063186769	0.062971934
0.527481851	1.86368833	0.021539972	0.040143794	0.66463235	0.6623726
0.807263134	0.726241527	0.276751969	0.200988772	3.32762868	3.316314743
0.55190323	1.834808541	0.028260355	0.051852341	0.858482468	0.855563628
0.612808637	1.688971541	0.052960157	0.089448199	1.480930442	1.475895278
0.573633704	1.794553082	0.035629355	0.063938769	1.058588895	1.054989693
0.624998494	1.647954515	0.059603783	0.098224323	1.626230512	1.620701328
0.404049408	1.739445056	0.004351178	0.007568635	0.125308527	0.124882478
…	…	…	…	…	…

We are now in a position to introduce the Markov Chain Monte Carlo (MCMC) method with the Metropolis algorithm (Metropolis *et al.*, 1953) which is a special case of the Metroplis-Hastings algorithm (Hastings, 1970). Our interest is the same as above, i.e., the posterior probability density function f(p|y). In short, the algorithm consists of three steps. First, we start with any p_0 satisfying $f(p_0)*f(y|p_0) > 0$. Second, we generate a new p* by using either one of two jumping functions: (1) a random walk chain or (2) an independent chain. For illustration of a random walk chain, we will use the following jumping function:

$$\begin{aligned} &z = Rnd()/10 \\ &p^* = p_0 + if[Rnd() > 0.5, z, -z] \end{aligned} \tag{13.69}$$

where z is the step length of the random walk, Rnd() is a random number generator generating random numbers between 0 and 1, and the division by 10 is to limit the step length so that the random walk will not be too erratic. We also need to constrain p* within the range of, say, (0.00001, 0.99999), i.e., when p* is set to 0.00001 when p* ≤ 0 and set to 0.99999 when p* ≥ 1. If p* When Rnd() > 0.5, p* is equal to p plus a step length, otherwise p* is p minus a step length. This symmetry in the jumping function is required by the Metropolis algorithm.

In the third step, we compute the ratio of

$$\alpha = \frac{f(p^*|y)}{f(p_0|y)} = \frac{f(p^*)f(y|p^*)}{f(p_0)f(y|p_0)} \tag{13.70}$$

Note that the integration in the denominator of Eq. (13.59) for f(p|y) cancels out when we compute the ratio α. This is the main advantage of the MCMC method.

If $\alpha < 1$, then we accept p* with a probability of α and reject p* with a probability of $(1 - \alpha)$. If $\alpha \geq 1$, then we accept p*. The accepted p* becomes p_1, and the procedure is repeated according to Eqs. (13.69) and (13.70), with p_0 replaced by $p_1, p_2, \ldots, p_n$.

The chain proceeds with no end. However, it is expected that, when t become sufficiently large (e.g., when t = k is typically a large number), the chain should approach its stationary distribution and samples from the vector $(p_{k+1}, p_{k+1}, \ldots, p_{k+n})$ are samples from f(p|y). The period from t = 0 to t = k is termed the burn-in period. Figure 13-9 is a plot of p_t versus t, started with $p_0 = 0.111$. The chain appears to have converged soon after t = 1000, with p_t values distributed roughly around 0.8 (Figure 13-9).

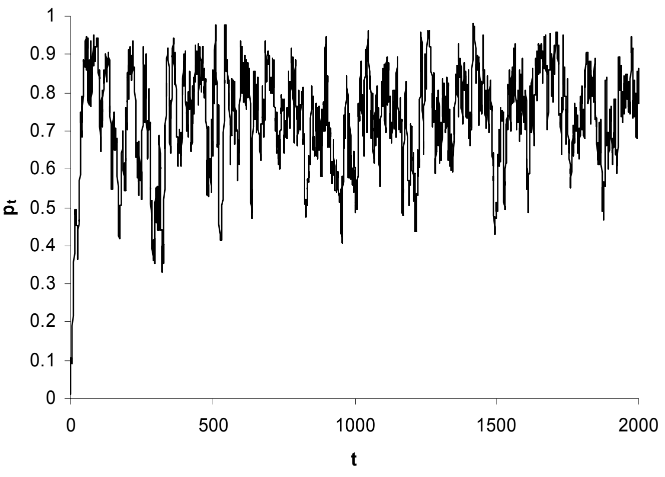


Figure 13-9. The dynamic behavior of accepted p values along a random walk chain using Metropolis algorithm.

Bayesian phylogenetics is obviously much more complicated than the one-parameter case we have illustrated. The parameters in Bayesian phylogenetics, often collectively designated as θ, is a collection of the tree topology, the rate matrix and the branch lengths, with the likelihood function formulated as in the maximum likelihood method. However, the main controversy remains to be the justification of the prior probability (Felsenstein, 2004; Pickett and Randle, 2005; Zwickl and Holder, 2004) which is problematic even for the simple example of estimating *p*.

In summary, the accuracy of phylogenetic reconstruction depends mainly on (1) the sequence quality, (2) the correct identification of homologous sites by sequence alignment, (3) regularity of the substitution processes, e.g., stationarity along different lineages, absence of heterotachy and little variation in the substitution rate over sites, (4) consistency, efficiency and little bias in the estimation method, e.g., not plagued by the long-branch attraction problem, and (5) sequence divergence, i.e., neither too conserved as to contain few substitutions nor too diverged as to experience substantial substitution saturation. Readers wish to do research in molecular phylogenetics and evolution should consult more detailed references (Felsenstein, 2004; Hillis *et al.*, 1996; Li, 1997; Nei and Kumar, 2000; Semple and Steel, 2003).

Chapter 14

FUNDAMENTALS OF PROTEOMICS
Peptide mass fingerprinting

1. INTRODUCTION

The terms “proteome” and “proteomics” were coined by Marc Wilkins and colleagues in 1994 (Ezzell, 2002), with proteomics referring specifically to studies of proteins using the method of mass spectrometry (MS). Proteomics has since become one of the two major components of what is now known as protein science (e.g., Lesk, 2004), with the other component being protein structure determination, typically by X-ray crystallography and nuclear magnetic resonance.

Mass spectrometry used in combination with affinity purification and/or chemical cross-linking has made significant contributions to protein interaction networks (Figeys, 2003b, 2003a; Vasilescu and Figeys, 2006). Protein arrays have recently been developed to directly assess protein-protein interactions (Figeys, 2002; Sloane *et al.*, 2002; Wilson and Nock, 2002). Ultimately, all proteins, isolated either by conventional 2D gel, protein arrays, or other affinity purification methods, have to go through MS for protein identification. For this reason, protein identification is the most fundamental in proteomics.

Modern large-scale protein identification is through protein mass fingerprinting (Sloane *et al.*, 2002; Washburn *et al.*, 2001; Yates, 2004a, 2004b), which is the main subject in this chapter. We will first give a brief overview of MS, followed by a few computational methods involved in peptide mass fingerprinting.

2. PROTEIN MASS SPECTROMETRY

All MS instrumentations include two essential components, an ionizer that generates gaseous ions from a sample and a mass analyzer that generates the number as well as the mass/charge ratios, i.e., m/z values of each type of ions. A computational protocol called charge deconvolution (or just deconvolution) uses the m/z values to produce the estimated molecular mass of the molecule of interest, e.g., protein or peptide. Deconvolution is explained in the next section.

Two types of ionizers are used frequently in proteomics (Lesk, 2004; Liebler *et al.*, 2002), the matrix-assisted laser desorption ionization (MALDI) and electrospray ionization (ESI). The MALDI ionizer generates predominantly singly charged ions (which may be positive or negative but generally only positive ions are analyzed by MS) and used most often in peptide mass fingerprinting because of its high accuracy in measuring peptide mass. An extension of the MALDI ionizer, termed surface-enhanced laser desorption/ionization or SELDI (Forde and McCutchen-Maloney, 2002; Tang *et al.*, 2004; Wright, 2002; Yip and Lomas, 2002) has recently been developed for identifying proteins of different sizes on protein arrays.

The ESI ionizer generates ions carrying different charges because peptides or proteins ionized by ESI ionizers are in aqueous solutions. Recall that proteins will carry a net charge when the pH of the solution differs from the protein isoelectric point (pI, review Chapter 10 if you have forgotten the computation of pI). They become more and more positively charged in acidic solutions and more and more negatively charged in basic solutions. For MS analysis, the peptides are typically ionized to carry positive charged in acidic solutions. Because a protein or a peptide may have multiple lysine, arginine and histidine residues, the resulting peptide or protein ions may consequently carry multiple charges. For example, a peptide with molecular mass of m may have n different types of charged ions from the ESI ionizer, designated as ion_1, ion_2, …, ion_n, carrying z_0, (z_0+1), (z_0+2), …, (z_0+n-1) positive charges (protons), respectively. Note that we have designated z_0 as the charge of the least-charged ion.

Each proton adds one atomic mass to the ion, so the actual molecular masses of ion_1, ion_2, …, ion_n carrying z_0, (z_0+1), (z_0+2), …, (z_0+n-1) protons are $(m+z_0)$, $(m+z_0+1)$, $(m+z_0+2)$, …, $(m+z_0+n-1)$, respectively. However, MS does not measure the molecular mass of the ions directly. Instead it outputs the m_{ion}/z_{ion} ratios where m_{ion} is the mass of the ion (i.e., m plus the total mass of extra protons) and z_{ion} is the positive charge of the ion (the number of the protons). Given the n types of differently charged ions, an ESI-MS will output $(m+z_0)/z_0$, $(m+z_0+1)/(z_0+1)$, $(m+z_0+2)/(z_0+1)$, …,

$(m+z_0+n-1)/(z_0+n-1)$. Charge deconvolution in the next section takes this output to estimate m and z_0.

Aside from an ionizer, a MS will always have a mass analyzer. Frequently used mass analyzers are time of flight (TOF), quadrupole or ion trap analyzers. Different MS instrumentations are often specified by the combination of the ionizer and the mass analyzer. For example, MALDI-TOF MS is a frequent combination for peptide mass fingerprinting after digesting proteins into small peptides, and SELDI-TOF MS is used typically for large-scale protein identification in protein arrays involving proteins over a wide mass range.

A mass analyzer has fixed measurement range of m/z values. If the maximum m/z range for a given mass analyzer is 2000, then the analyzer can measure the mass of a peptide up to the molecular mass of 2000 when ions are singly charged, as is the case with MALDI ionizer. If ions carry multiple charges, as in the case with ESI ionizers, then the same mass analyzer can be used to measure the molecular mass of much larger peptides or even entire proteins. For example, if the ion mass is 10000 but it carries 10 positive charges, then its m/z ratio is only about 1000, well within the measurement range of the mass analyzer. For this reason, MS with an ESI ionizer is able to measure the molecular mass of much larger molecules than that with a MALDI ionizer.

There are excellent descriptions of MS hardware in the proteomic framework (e.g., Liebler *et al.*, 2002) for readers who are interested in MS hardware. I will focus only on what is important but missing in other books on proteomics.

3. CHARGE DECONVOLUTION

Deconvolution in MS literature refers to the protocol of computing the molecular mass from the distribution of multiply charged ions of the molecule of interest. Such multiply charged ions are typical in MS with an ESI ionizer. MS data obtained with a MALDI ionizer do not need charge deconvolution because ions are predominantly singly charged and the peptide mass can be derived directly as the m/z ratio minus the proton mass (which is 1).

Let us start with a simple example taken from Liebler (2002, p. 67). An ESI-MS analysis of a peptide revealed two ions, ion_1 with a $m_{ion}/z_{ion} = 784.7$ and ion_2 with a $m_{ion}/z_{ion} = 1567.9$. How to estimate the molecular mass (m) of the peptide?

There are two categories of deconvolution methods, one being probabilistic and the other deterministic. Here we employ only a representative of the deterministic method because it is simpler and in most cases sufficient.

Recall that we have designated z_0 as the charge of the least-charged ion, which is ion$_2$ in this example (Note that, with the same m, the more charge, the smaller the m/z ratio). Ion$_1$ then carries (z0+1) protons. Each proton adds one atomic mass to the peptide, so the expected m/z values for ion$_1$, and ion$_2$, designated as mz$_1$, mz$_2$, respectively, can be expressed as

$$\begin{aligned} mz_1 &= (m + z_0 + 1)/(z_0 + 1) \\ mz_2 &= (m + z_0)/z_0 \end{aligned} \tag{14.1}$$

The least-square estimation of the two parameters (i.e., m and z_0) is to minimize the sum of squared deviation (SS) between the observed mz$_i$ values, i.e., 784.7 and 1567.9, and their respectively expected mz$_1$ and mz$_2$ in Eq. (14.1):

$$\begin{aligned} SS &= (mz_1 - 784.7)^2 + (mz_2 - 1567.9)^2 \\ &= \left(\frac{m + z_0 + 1}{z_0 + 1} - 784.7 \right)^2 + \left(\frac{m + z_0}{z_0} - 1567.9 \right)^2 \end{aligned} \tag{14.2}$$

To minimize SS, we take partial derivative of SS with respect to m and z_0, set the two partial derivatives to 0 and solve for m and z_0. The two partial derivatives are:

$$D_m = \frac{\partial SS}{\partial m} = \frac{2\left(\frac{m + z_0 + 1}{z_0 + 1} - 784.7 \right)}{z_0 + 1} + \frac{2\left(\frac{m + z_0}{z_0} - 1567.9 \right)}{z_0} \tag{14.3}$$

$$\begin{aligned} D_{z_0} = \frac{\partial SS}{\partial z_0} &= 2\left(\frac{m + z_0 + 1}{z_0 + 1} - 784.7 \right)\left(\frac{1}{z_0 + 1} - \frac{m + z_0 + 1}{(z_0 + 1)^2} \right) \\ &+ 2\left(\frac{m + z_0}{z_0} - 1567.9 \right)\left(\frac{1}{z_0} - \frac{m + z_0}{z_0^2} \right) \end{aligned} \tag{14.4}$$

Setting D$_m$ and D$_{z0}$ to 0 and solving the simultaneous equations with the constraint of $z_0 > 0$ result in m = 1567.9032 and z_0 = 1.0006 (which is taken to mean 1). This means that ion$_1$ carries two (=z_0+1) protons and ion$_2$ carries only one proton.

One may ask why we can't just assume $z_0 = 1$, so that the m_{ion}/z_{ion} for ion$_2$ (= 1567.9) can be taken as a direct measure of m. The reason is that z_0 is often not 1. For long peptides or proteins, the chance of getting singly charged ion in an ESI-MS instrumentation is typically quite small. It might help to introduce a numerical illustration.

Amino acid residues that may carry charges are mainly the three basic amino acids (arginine, lysine, histidine) and two acidic amino acids (glutamic acid and aspartic acid), plus the N-terminal amino acid with an amino group and C-terminal amino acid with a carboxyl group. In acidic solutions (say pH = 3), the probability of the amino group of the three basic amino acids being protonated is nearly 1 according to the following equation (see Chapter 10 for its derivation).

$$P_{\mathrm{NH}_3^+} = \frac{1}{10^{pH-pK_\mathrm{a}}+1} \qquad (14.5)$$

You may substitute pH = 3 and the pK$_a$ value (12.50, 10.79 and 6.50 for arginine, lysine and histidine, respectively, Table 10-1) into the equation to verify that the proportion is nearly 1. The probability of the amino group in the N-terminal amino acid, with its pK$_a$ = 8.56, being protonated is also nearly 1.

The probability of the two acidic amino acids being protonated can be calculated according to the following equation:

$$P_{\mathrm{RCOO}^-} = \frac{10^{pH-pK_\mathrm{a}}}{1+10^{pH-pK_\mathrm{a}}} \qquad (14.6)$$

Now suppose a protein contains 10 lysine residues and no arginine, histidine, aspartic acid and glutamic acid residues. In a solution with pH = 3, the 10 lysine residues are protonated, so is the amino group of the N-terminal amino acid. The C-terminal carboxyl has a probability of 0.21595 being protonated (You may verify this by substituting pK$_a$ = 3.56 for the C-terminal carboxyl into the equation above). With N copies of such a protein in the solution, There will be only two ions, one with its C-terminal carboxyl protonated and the other not, with their respectively proportions being 0.21595 and (1-0.21595). This implies that the net charges of the two ions will be 11 and 10, respectively. There will essentially be no ion carrying fewer than 10 charges. In this case z_0 = 10.

One may argue that our example is too artificial, with a protein carrying no negatively charged residues such as aspartic acid or glutamic acid residues. We will now consider the case when the protein not only contain 10 lysine residues, but also 10 aspartic acid residues (but no arginine, histidine, and glutamic acid residues as before). The proportion of aspartic acid residues (whose pKa = 3.91) being protonated is 0.10955 according to Eq. (14.6). The frequency distribution of the proteins with 0, 1, …, 10 aspartic acid residues protonated is then specified by the binomial distribution of $(p + q)^{10}$, where p = 0.10955 and q = 1 - p. Thus, given N copies of such a protein in the solution, the proportion of proteins with 0, 1, …, 10 aspartic acid residues protonated is 0.31340, 0.38557, 0.21346, 0.07003 0.01508, 0.00223, 0.00023, 0.00002, 0.00000, 0.00000, and 0.00000, respectively. The proportion is also the proportion of protein ions carrying 10, 9, …, 0 extra protons if we ignore the N-terminal and C-terminal amino acids (You may take into consideration the N-terminal and C-terminal amino acids as an exercise). Obviously, the chance of having a protein ion carrying five extra protons is already very small (= 0.00223), and the chance of having an ion carrying fewer than three extra protons is essentially zero. Thus, the chance of having $z_0 = 1$ is often negligibly small, and we consequently cannot assume $z_0 = 1$. In short, it is necessary to estimate both z_0 and m.

Now that you are convinced that we cannot assume $z_0 = 1$, we will introduce a more complicated example involving a long peptide, with the output from ESI-MS shown in Table 14-1. There are 12 types of differentially charged ions with their respective m/z ratios. The estimated molecular mass of the molecule is about 12358 and the estimated z is listed in the last column of Table (14-1). How did we get such estimates?

Table 14-1. Output from ESI MS with estimated z in the last column.

Ion	m/z	Number	z
1	687.72	1000	18
2	727.91	4600	17
3	773.39	9000	16
4	824.93	13400	15
5	883.74	13400	14
6	951.67	10000	13
7	1030.88	8000	12
8	1124.46	7500	11
9	1236.91	6500	10
10	1374.16	6000	9
11	1545.66	5600	8
12	1766.29	1000	7

We first will ignore the column headed by "Number" and use only information in the column headed by "m/z". Again recall that z_0 is the number of charges (i.e., extra protons) carried by the least charged ion, which is ion_{12} with its m/z = 1766.29. Ion_{11}, ion_{10}, …, and ion_1 carry (z_0+1), (z_0+2), … and (z_0+11) protons, respectively. Designate m as the mass of the peptide. Because the molecular mass of each proton is one, the actual mass of ion_{12}, ion_{11}, …, and ion_1 is $(m + z_0)$, $(m + z_0 + 1)$, …, $(m + z_0 + 11)$. Thus defined, the expected m/z values for ion_1, ion_2, …, ion_{12}, designated as mz_1, mz_2, …, mz_{12}, respectively, can be expressed as

$$\begin{aligned} mz_1 &= (m + z_0 + 11)/(z_0 + 11) \\ mz_2 &= (m + z_0 + 10)/(z_0 + 10) \\ &\dots \\ mz_{12} &= (m + z_0 + 0)/(z_0 + 0) \end{aligned} \qquad (14.7)$$

The least-square estimation of the two parameters (i.e., m and z_0) is to minimize the sum of squared deviation between the observed m/z values (i.e., 687.72, 727.91, etc., in Table 14-1) and the expected mz_1, mz_2, etc., in Eq. (14.7):

$$SS = (mz_1 - 687.72)^2 + (mz_2 - 727.91)^2 + \dots + (mz_{12} - 1766.29)^2 \qquad (14.8)$$

To minimize SS, one naturally would take partial derivatives of SS with respect to m and z_0 to obtain D_m and D_{z0}, set them to 0 and solve the simultaneous equations to obtain m and z_0, just as we have done before with only two ions. However, with 12 ions, D_m and D_{z0} may become too complicated for analytically solutions of m and z_0 to be obtained. So it is time to learn how to obtain numerical solutions by numerical iteration.

We can substitute different values of m and z0 to obtain D_m and D_{z0} values. The m and z_0 values that make D_m and D_{z0} values closest to 0 are the best estimates. Table 14-2 shows such an iteration process.

Table 14-2. Estimation of m and z_0 by computer iteration.

m	z_0	$D_{m'}$	D_{z0}
12350	7	-1.61	2012.45
12360	7	0.38	-533.41
12370	7	2.37	-3083.39
12365	7	1.38	-1807.89
12361	7	0.58	-788.22
12359	7	0.18	-278.64
12358	7	-0.02	-23.91
12357	7	-0.22	230.78
12358	6	323.38	-512885.17
12358	8	-206.63	251562.87

We first tried m values from 12350 to 12370 and z_0 = 7, and we found the m value equal to 12358 to yield the smallest D_m and D_{z0} values (Table 14-2). Then we fix m = 12358 but vary z_0 values from 6 to 8, and found both z_0 = 6 and z_0 = 8 to be very poor estimates, resulting in very large D_m and D_{z0} values (Table 14-2). Thus, we conclude that our z_0 = 7, and the z values in the last column of Table 14-1 are easily obtained because $z_{12} = z_0$, $z_{11} = z_0 + 1$, $z_{10} = z_0 + 2$, …, z1 = z_0 + 11. Of course one can continue the iteration process to obtain more accurate estimate of m.

More advanced algorithms would incorporate the column headed by "Number" in Table 14-1 into a weighted estimation. However, this is often not necessary given the outstanding performance of modern MS.

One may ask what we should do when some of the ions are missing, e.g., ion_6 may not get ionized and consequently will not have its m/z value reported. One can formulate more advanced probabilistic methods to evaluate the probability of missing ions. However, a simpler method is just to plot the observed m/z ratio versus the series of n, n-1, n-2, …, 1 where n is the number of types of differently charged ions (e.g., 12 in our example in Table 14-1). Such a plot, which I call missing-ion plot, for data in Table 14-1 is shown in Figure 14-1a.

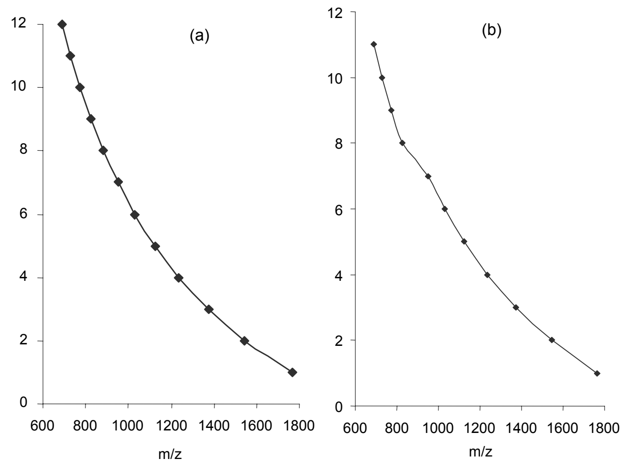


Figure 14-1. Missing-ion plot, with one ion missing in (b).

The curve is smooth when no ion is missing (Figure 14-1). In contrast, when one ion is missing (e.g., when ion_6 in Table 14-1 with m/z = 951.67 is missing), the curve is no longer smooth (Figure 14-1b). The curve would become even more twisted if two consecutive ions are missing (by two consecutive ions I mean two ions carrying i and i+1 charges, respectively).

Modern MS data are so accurate that a missing ion can generally be identified by such a plot.

How many positively-charged ions we should expect to have, given a peptide "DAFLGSFLYEYSR"? The last R (arginine residue) and the N-terminal amino group can both be protonated. So we should have only two different positively-charged ions, one with $z = 1$ and the other with $z = 2$. This is in fact the peptide that provides us with the first example in this section where we have ion_1 with a $m_{ion}/z_{ion} = 784.7$ and ion_2 with a $m_{ion}/z_{ion} = 1567.9$.

4. PEPTIDE MASS FINGERPRINTING

Peptide mass fingerprinting (PMF) is for protein identification. For example, one may perform 2D-SDS-PAGE of liver proteins between a normal person and a liver cancer patient. Comparing the two gels, one may find a dot that is different between the two. One naturally wishes to know what protein the dot represents. Establishing a link from a protein dot on the gel to a protein-coding gene on the genome is where PMF shines. Below I detail the four essential steps in PMF.

4.1 Peptide digestion

The first step in PMF is to cut out the protein dot on the gel and digest it into peptides by using one of proteases (Table 14-3). For example, trypsin cuts after Arg and Lys residues when the residue is not immediately followed by a Pro (Table 14-3). The purpose of including amino acid frequencies in Table 14-3 is to correct a misconception. Some authors (e.g., Liebler *et al.*, 2002, p. 52) have remarked that chymotrypsin may cleave too frequently because it cleaves at three amino acid residues (Phe, Trp and Try) to yield too many peptides that are too small to be informative in MS analysis. However, for human proteins, trypsin is expected to cut much more frequently than chymotrypsin because Arg and Lys account for a total of 11.58% of all amino acid residues, whereas Phe, Trp and Tyr jointly account for only 7.30% of all amino acid residues (Table 14-3). This implies that trypsin will cleave the proteins into much smaller peptides than chymotrypsin.

Trypsin is widely used in protein digestion in MS analysis. However, it is not suitable for all proteins. For example, the human *DEXI* gene codes for 95 amino acids which include only one Arg and no Lys residue. The protein consequently cannot produce suitable peptides for MS analysis with trypsin digestion. However, 13 of its residues are Phe, Trp and Tyr and it can consequently be cut by chymotrypsin into peptides suitable for MS analysis. It also contains nine Glu residues and can be cut by Glu C into peptides suitable for MS analysis. On the other hand, the human *PRB3* gene codes 351 amino acid residues but contains only one Glu residue and no Phe, Trp or Tyr residue, i.e., Glu C will cut it only once and chymotrypsin will not cut it at all. However, it contains 17 Arg and 17 Lys residues and can be cut by trypsin into peptides with lengths well suited for MS analysis. A large-scale peptide mass fingerprinting will almost always involve digestion with more than one protease.

Table 14-3. Amino acid frequencies from 34179 annotated human CDSs from GenBank and protease cleavage site. X – cut after the specific amino acid; \Pro – cleavage inhibited if the cleavage site is followed by proline.

AA	Percent	Trypsin	Chymotrypsin	Asp N	Glu C	Lys C
Ala	7.21					
Arg	5.96	X\Pro				
Asn	3.48					
Asp	4.65			X		
Cys	2.29					
Gln	4.75					
Glu	7.02				X\Pro	
Gly	6.84					
His	2.60					
Ile	4.19					
Leu	9.77					
Lys	5.62	X\Pro				X\Pro
Met	2.10					
Phe	3.51		X\Pro			
Pro	6.65					
Ser	8.34					
Thr	5.33					
Trp	1.25		X\Pro			
Tyr	2.54		X\Pro			
Val	5.89					
Sum	100	11.58	7.30	4.65	7.02	5.62

It is almost always a good practice to have a rough estimate of the average peptide length as well as the distribution of the peptide lengths after digestion with a certain protease. For example, suppose we digest a human protein with trypsin, which cleaves the protein after the Lys and Arg residue when they are not followed by a Pro. From Table 14-3, we know that Lys and Arg residues (hereafter referred to as KR residues) jointly account for 11.58% of the amino acid residues of human proteins and Pro accounts for 6.65%, which is also the probability that a KR residue is followed by a Pro.

Thus, the probability that a KR residue that is not followed by a Pro (i.e., the probability of cleavage by trypsin) is

$$p = 0.1158(1-0.0665) = 0.1081 \tag{14.9}$$

The distribution of the peptide length (*l*) follows the geometric distribution:

$$P(L = l \mid p) = (1-p)^{l-1}\, p, \tag{14.10}$$

The expected mean and variance of the peptide length are then, respectively,

$$\begin{aligned} E(L) &= \frac{1}{p} = 9.25 \\ Var(L) &= \frac{(1-p)}{p^2} = 76.32 \end{aligned} \tag{14.11}$$

In contrast, digesting the same protein with chymotrypsin would generate peptides with an expected mean length of 14.67 and an expected variance of the peptide lengths equal to 200.67.

In the genome of the mammalian gastric pathogen, *Helicobacter pylori* (strain 26695), the proportions of Arg, Lys and Pro are 0.0345, 0.0894 and 0.0328, respectively, based on its 1576 annotated proteins. If we have a representative sample of *H. pylori* proteins and use trypsin to completely digest all these proteins, what is the expected mean length of the resulting peptides?

From the three proportions for Arg, Lys and Pro, we can estimate p = (0.0345+0.0894)*(1-0.0328) = 0.119836. Thus, the expected mean and variance of the peptide lengths, after complete trypsin digestion, are 8.3447 and 61.2898, respectively. The observed distribution, based on an *in silico* trypsin digestion of a random sample of *H. pylori*, is close to the expected distribution (Figure 14-2). Generally peptides of length between 5 and 21 residues can be measured accurately by MALDI-TOF MS. The distribution in Figure 14-2 has a proportion of 0.504300196 of the peptides with lengths between 5 and 21. This means that about half of the digested peptides from *H. pylori* proteins can be measured by MALDI-TOF MS with high accuracy.

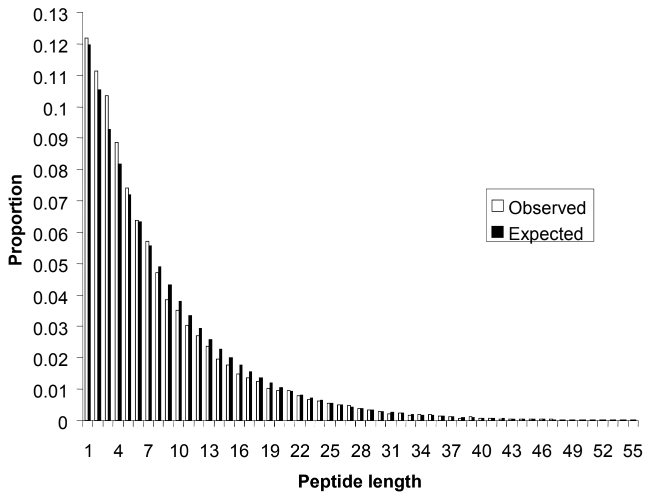


Figure 14-2. Observed and expected distribution of peptide length, from trypsin digestion of a random sample of proteins in *H. pylori* Str 26695.

4.2 MS determination of peptide mass

Now that the protein dot has been reduced to a number of peptides (referred to hereafter as query peptides), we proceed to the second step in PMF by subjecting these query peptides to MS to obtain peptide masses. Suppose we now have determined the peptide mass of n peptides resulting from digesting the protein dot from the 2-D gel. We typically will create a file, say ProteinDot1.txt, to store these n peptide masses, m_1, m_2, …, m_n. For example, one protein dot from a sample of *H. pylori* proteins could have eight peptides with their molecular masses determined by MS:

```
832.94
974.05
1105.16
1526.68
1537.92
1653.86
1680.87
2231.42
```

The list of eight peptide masses, hereafter referred to as query peptide masses, will be matched against the peptide mass of all possible peptides from an *in silico* digestion of all *H. pylori* proteins to identify the protein. This brings us to the third step of creating a database of peptide masses.

4.3 Protein database and *in silico* digestion

The third step of PMF is to obtain a relevant database of proteins and perform an *in silico* digestion, using the same protease that has been used to digest the protein dot. If one is working on human, then the relevant database of proteins would obviously be human proteins. For example, one may retrieve all 34169 annotated human CDSs, translate them into proteins and perform an *in silico* digestion to obtain all possible peptides with their respective masses. We have already learned how to compute the molecular mass of a peptide in Chapter 11.

If one is working on *H. pylori,* then one may retrieve the annotated protein-coding genes in sequenced *H. pylori* genomes and perform *in silico* digestion to obtain all possible peptides with their respectively masses. The *in silico* trypsin digestion of all 1576 annotated proteins in *H. pylori* strain 26695 generates 61160 peptides, which are partially shown in Table 14-4.

The list of peptide masses from the database will be referred to as database peptide masses (designated by M_j) to distinguish them from the query peptide masses (designated by m_i) which pertain to peptides from an isolated protein (e.g., a protein dot in a 2D-SDS-PAGE).

4.4 Protein identification

The fourth and final step in PMF is to search each of the query peptide masses (e.g., the eight query peptide masses from a *H. pylori* protein dot in Section 4.2) against *H. pylori* database peptide masses (Table 14-4). For example, the first query peptide mass, 832.94, matches five database peptide masses (Table 14-4), the second query peptide mass, 974.05, matches six database peptide masses, and so on. The number of matches depends on the accuracy of MS. If peptide mass m_i is accurate to 0.5 Dalton, then any database peptide within the peptide mass range of (m_i – 0.5, m_i + 0.5) would be considered as a match.

We note that all eight query peptide masses match peptides from the protein with gene index (GeneInd) of 319 (Table 14-4). We can therefore conclude that the query peptides come from the gene with GeneInd = 319, which has a locus_tag of HP0324 and is described as “outer membrane protein (omp10)” in the GenBank file (NC_000915). There is no other gene that comes close to this gene in term of the number of matches.

Table 14-4. Partial list of peptide masses from trypsin digestion of the 1576 annotated proteins in *H. pylori* strain 26695. GeneInd specifies the gene from which the peptide is from. Note that the peptide at the C-terminal of the protein may not end with K (Lys) or R (Arg).

Mass	GeneInd	Peptide
832.934	38	QFTYFK
832.941	48	NNFPTLK
832.941	319	FPTINNK
832.941	476	GFNAPSLK
832.962	106	LGEAMANK
…		
974.023	853	GVEAEVQDK
974.052	319	YYTTDALK
974.052	507	VYDLSSYK
974.066	57	ITNEQIEK
974.066	171	TALVENEAK
974.066	1235	YSDLALHR
…		
1105.114	5	DDDNLALSSR
1105.155	319	EESAAPSWTK
1105.184	493	VDYNYYLR
…		
1526.644	87	GLDEAIEFLEEYV
1526.684	319	VFAFYVGYNYHF
1526.688	1386	SLGNNLLYNTYVR
…		
1537.808	71	GVAFSLLSFLEGGLK
1537.919	319	MLVGASLLTHALIAK
1538.613	747	FFDLGEYFEEDK
…		
1653.84	852	IHFAQNYQLFSSAK
1653.856	319	YFAFLDWQGYGMR
1653.861	1379	MIGGSENIESAISFAK
1653.87	510	EIVAYLDEYIIGQK
…		
1680.817	517	VPNQATFYDDLQAAK
1680.868	242	IGLNQQEIDAIQNPK
1680.868	319	ALFVDEHEFEIGFK
1680.883	1360	MDEVDLIFEEEAIK
…		
2230.723	1277	FYATLALSCVFLTITNILVK
2231.42	319	EVTNYQTGYTNIITSVNSVK
2231.42	1503	DFYEELLYILGLEEQNDK
2231.499	208	EVDVLGGAMGIITDHSGLQYR

The intuitive argument above linking a protein dot and a protein-coding gene on the genome has problems, especially in eukaryotes where proteins differ dramatically in lengths. For example, the mammalian dystrophin protein coded by the *dmd* gene contains about 4000 amino acid residues coded by about 75 exons. It consequently would have a much large probability (100 times to be exact) of getting a random match than a small protein of 40 amino acid residues. We therefore need to develop a statistical framework to help us decide whether our identification is correct or not.

We have two hypotheses, i.e., the protein dot is HP0324 (designated as θ_{Yes}) and it is not (designated as θ_{No}). Designate Y_{ij} as the event of query peptide mass m_i matching database peptide mass M_j of HP0324 (19 peptides are generated by *in silico* digestion of the annotated HP0324 protein, so j = 1 2, …, 19). The likelihood of Y_{ij}, i.e., the probability of Y_{ij} happening given θ_{Yes} is true, is generally close to 1 (although molecular mechanisms such as posttranslational modification may reduce this value slightly), i.e.,

$$p(Y_{ij} \mid \theta_{Yes}) \approx 1 \tag{14.12}$$

If the protein dot is not HP0324, then what is the probability of event Y_{ij} happening? This can be estimated empirically by searching the 17 M_j values from HP0324 against the rest of the M values from other *H. pylori* proteins. Designate the number of matches for M_1, M_2, …, M_{17} as N_1, N_2, …, N_{17}, respectively, we can estimate

$$p(Y_{ij} \mid \theta_{No}) = \frac{\sum_{j=1}^{17} N_j}{17(N_T - 17)} \tag{14.13}$$

where N_T is the total number of database peptides results from *in silico* digestion. For simplicity, let's assume that $p(Y_{ij}|\theta_{No}) = 0.0003$.

Now designating the protein length of HP0324 as L_{HP0324}, and the total length of all 1576 *H. pylori* proteins as L_T, we have the prior probabilities for θ_{Yes} and θ_{No} as

$$\begin{aligned} p(\theta_{Yes}) &= \frac{L_{HP0324}}{L_{Total}} = \frac{255}{503015} = 0.0005 \\ p(\theta_{No}) &= 1 - p(\theta_{Yes}) = 0.9995 \end{aligned} \tag{14.14}$$

According to Bayes' theorem, the probability that θ_{No} is true is

$$\begin{aligned} P(\theta_{No} \mid Y_{ij}) &= \frac{P(Y_{ij} \mid \theta_{No})P(\theta_{No})}{P(Y_{ij} \mid \theta_{Yes})P(\theta_{Yes}) + P(Y_{ij} \mid \theta_{No})P(\theta_{No})} \\ &= \frac{0.0003 \times 0.9995}{1 \times 0.0005 + 0.0003 \times 0.9995} \approx 0.37488 \end{aligned} \quad (14.15)$$

Thus, with only one Y_{ij} event, we cannot reject θ_{No}. However, we have eight m_i values that all match M_j values from HP0324. So the final probability that θ_{No} is true is $0.37488^8 = 0.0004$. So θ_{No} can be conclusively rejected and we conclude that the protein dot is indeed HP0324. The formulation above can be further refined by taking into consideration the fact that peptides of different lengths have different matching probabilities against database peptide mass.

Peptide mass fingerprinting, together with quantification of protein abundance, ultimately leads to two types of data for further bioinformatics analysis. The first type is between the control and the experiment and the second type is what is known as time-course data, obtained by sampling the proteome of a cell lineage over different time points. For example, one may synchronize the development cycle of yeast cells and sampling the proteome at regular time intervals during the yeast cell cycle, or trigger the developmental cascade of a stem cell lineage and sample the proteome at regular time intervals. These two types of data parallel those from transcriptomic experiments and can be analyzed similarly to identify genes that are up-regulated or down-regulated at the protein level.

REFERENCES

Adachi, J., and Hasegawa, M., 1996, Model of amino acid substitution in proteins encoded by mitochondrial DNA. *J. Mol. Evol.* **42**: 459-468.

Aerts, S., Van Loo, P., Thijs, G., Mayer, H., de Martin, R., Moreau, Y., and De Moor, B., 2005, TOUCAN 2: the all-inclusive open source workbench for regulatory sequence analysis. *Nucleic Acids Res* 33: W393-396.

Agris, P.F., 2004, Decoding the genome: a modified view. *Nucleic Acids Res* 32: 223-238.

Aird, W.C., Parvin, J.D., Sharp, P.A., and Rosenberg, R.D., 1994, The interaction of GATA-binding proteins and basal transcription factors with GATA box-containing core promoters. A model of tissue-specific gene expression. *J. Biol. Chem.* 269: 883-889.

Akashi, H., 1995, Inferring weak selection from patterns of polymorphism and divergence at "silent" sites in Drosophila DNA. *Genetics* 139: 1067-1076.

Akashi, H., 1997, Codon bias evolution in Drosophila. Population genetics of mutation-selection drift. *Gene* 205: 269-278.

Akashi, H., 2003, Translational selection and yeast proteome evolution. *Genetics* 164: 1291-1303.

Allen, A., Flemstrom, G., Garner, A., and Kivilaakso, E., 1993, Gastroduodenal mucosal protection. *Physiol. Rev.* 73: 823-857.

Alm, R.A., Bina, J., Andrews, B.M., Doig, P., Hancock, R.E., and Trust, T.J., 2000, Comparative genomics of *Helicobacter pylori*: analysis of the outer membrane protein families. *Infect. Immun.* 68: 4155-4168.

Altschul, S.F., Gish, W., Miller, W., Myers, E.W., and Lipman, D.J., 1990, Basic local alignment search tool. *J. Mol. Biol.* 215: 403-410.

Altschul, S.F., 1996, Local alignment statistics. *Meth. Enzymol.* 274: 460-480.

Altschul, S.F., Madden, T.L., Schaffer, A.A., Zhang, J., Zhang, Z., Miller, W., and Lipman, D.J., 1997, Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Res.* 25: 3389-3402.

Andachi, Y., Yamao, F., Muto, A., and Osawa, S., 1989, Codon recognition patterns as deduced from sequences of the complete set of transfer RNA species in Mycoplasma capricolum. Resemblance to mitochondria. *J Mol Biol* 209: 37-54.

Anderson, K.P., Crable, S.C., and Lingrel, J.B., 1998, Multiple Proteins Binding to a GATA-E Box-GATA Motif Regulate the Erythroid Kruppel-like Factor (EKLF) Gene. *J. Biol. Chem.* 273: 14347-14354.

Andersson, J.O., and Andersson, S.G., 1999, Genome degradation is an ongoing process in Rickettsia. *Molecular Biology & Evolution* 16: 1178-1191.

Andersson, S.G., Zomorodipour, A., Andersson, J.O., et al., 1998, The genome sequence of *Rickettsia prowazekii* and the origin of mitochondria. *Nature* 396: 133-140.

Aris-Brosou, S., 2003, How Bayes tests of molecular phylogenies compare with frequentist approaches. *Bioinformatics* 19: 618-624.

Aris-Brosou, S., and Yang, Z., 2003, Bayesian models of episodic evolution support a late precambrian explosive diversification of the Metazoa. *Mol Biol Evol* 20: 1947-1954.

Aris-Brosou, S., 2005, Determinants of adaptive evolution at the molecular level: the extended complexity hypothesis. *Mol Biol Evol* 22: 200-209.

Arnqvist, G., 2006, Sensory exploitation and sexual conflict. *Philos Trans R Soc Lond B Biol Sci* 361: 375-386.

Ast, G., 2004, How did alternative splicing evolve? *Nat Rev Genet* 5: 773-782.

Aylon, Y., and Kupiec, M., 2004, DSB repair: the yeast paradigm. *DNA Repair (Amst)* 3: 797-815.

Ayoubi, P., Jin, X., Leite, S., et al., 2002, PipeOnline 2.0: automated EST processing and functional data sorting. *Nucleic Acids Res* 30: 4761-4769.

Baik, S.C., Kim, K.M., Song, S.M., et al., 2004, Proteomic analysis of the sarcosine-insoluble outer membrane fraction of *Helicobacter pylori* strain 26695. *J. Bacteriol.* 186: 949-955.

Baldi, P., and Brunak, S., 2001, *Bioinformatics: the machine learning approach.* The MIT Press, Cambridge, Massachusetts.

Baliga, N.S., Pan, M., Goo, Y.A., et al., 2002, Coordinate regulation of energy transduction modules in Halobacterium sp. analyzed by a global systems approach. *Proc. Natl. Acad. Sci. U S A* 99: 14913-14918.

Baron, D., Cocquet, J., Xia, X., Fellous, M., Guiguen, Y., and Veitia, R.A., 2004, An evolutionary and functional analysis of FoxL2 in rainbow trout gonad differentiation. *J. Mol. Endocrinol.* 33: 705 - 715.

Barrell, B.G., Anderson, S., Bankier, A.T., et al., 1980, Different pattern of codon recognition by mammalian mitochondrial tRNAs. *Proc. Natl. Acad. Sci. U S A* 77: 3164-3166.

Bateman, A., Birney, E., Durbin, R., Eddy, S.R., Finn, R.D., and Sonnhammer, E.L., 1999, Pfam 3.1: 1313 multiple alignments and profile HMMs match the majority of proteins. *Nucleic Acids Res* 27: 260-262.

Bateman, A., Coin, L., Durbin, R., et al., 2004, The Pfam protein families database. *Nucleic Acids Res* 32: D138-141.

Bauerfeind, P., Garner, R., Dunn, B.E., and Mobley, H.L., 1997, Synthesis and activity of *Helicobacter pylori* urease and catalase at low pH. *Gut* 40: 25-30.

Baumgartner, H.K., and Montrose, M.H., 2004, Regulated alkali secretion acts in tandem with unstirred layers to regulate mouse gastric surface pH. *Gastroenterology* 126: 774-783.

Baxevanis, A.D., and Ouellette, B.F.F., 2005, *Bioinformatics: a practical guide to the analysis of genes and proteins.* Wiley, Hoboken, New Jersey.

Bebenek, K., Roberts, J.D., and Kunkel, T.A., 1992, The effects of dNTP pool imbalances on frameshift fidelity during DNA replication. *J. Biol. Chem.* 267: 3589-3596.

Beletskii, A., and Bhagwat, A.S., 1996, Transcription-induced mutations: increase in C to T mutations in the nontranscribed strand during transcription in Escherichia coli. *Proc. Natl. Acad. Sci. U S A* 93: 13919-13924.

Beletskii, A., and Bhagwat, A.S., 1998, Correlation between transcription and C to T mutations in the non-transcribed DNA strand. *Biol. Chem.* 379: 549-551.

Beletskii, A., Grigoriev, A., Joyce, S., and Bhagwat, A.S., 2000, Mutations induced by bacteriophage T7 RNA polymerase and their effects on the composition of the T7 genome. *J. Mol. Biol.* 300: 1057-1065.

Beletskii, A., and Bhagwat, A.S., 2001, Transcription-induced cytosine-to-thymine mutations are not dependent on sequence context of the target cytosine. *J. Bacteriol.* 183: 6491-6493.

Bennetzen, J.L., and Hall, B.D., 1982, Codon selection in yeast. *J. Biol. Chem.* 257: 3026-3031.

Benson, D.A., Karsch-Mizrachi, I., Lipman, D.J., Ostell, J., and Wheeler, D.L., 2005, GenBank. *Nucleic Acids Res* 33: D34-38.

Bentham, M., Mazaleyrat, S., and Harris, M., 2006, Role of myristoylation and N-terminal basic residues in membrane association of the human immunodeficiency virus type 1 Nef protein. *J Gen Virol* 87: 563-571.

Berg, J.M., Tymoczko, J.L., and Stryer, L., 2002, *Biochemistry.* W. H. Freeman and Co., New York.

Berg, O.G., and Martelius, M., 1995, Synonymous substitution-rate constants in Escherichia coli and Salmonella typhimurium and their relationship to gene expression and selection pressure. *J. Mol. Evol.* 41: 449-456.

Berg, O.G., 1996, Selection intensity for codon bias and the effective population size of Escherichia coli. *Genetics* 142: 1379-1382.

Besemer, J., and Borodovsky, M., 2005, GeneMark: web software for gene finding in prokaryotes, eukaryotes and viruses. *Nucleic Acids Res* 33: W451-454.

Bestor, T.H., and Coxon, A., 1993, The pros and cons of DNA methylation. *Curr. Biol.* 6: 384-386.

Bickel, D.R., 2003, Robust cluster analysis of microarray gene expression data with the number of clusters determined biologically. *Bioinformatics* 19: 818-824.

Birnboim, H.C., Sederoff, R.R., and Paterson, M.C., 1979, Distribution of polypyrimidine. polypurine segments in DNA from diverse organisms. *Eur J Biochem* 98: 301-307.

Bittner, M., Meltzer, P., and Trent, J., 1999, Data analysis and integration: of steps and arrows [news; comment] [see comments]. *Nat. Genet.* 22: 213-215.

Bogenhagen, D.F., and Clayton, D.A., 2003, The mitochondrial DNA replication bubble has not burst. *Trends Biochem Sci* 28: 357-360.

Bonen, L., 2006, Mitochondrial genes leave home. *New Phytol* 172: 379-381.

Bonen, L., and Calixte, S., 2006, Comparative analysis of bacterial-origin genes for plant mitochondrial ribosomal proteins. *Mol Biol Evol* 23: 701-712.

Borodovsky, M., and McIninch, J., 1993a, GENMARK: parallel gene recognition for both DNA strands. *Comput. Chem.* 17: 123-133.

Borodovsky, M., and McIninch, J., 1993b, Recognition of genes in DNA sequence with ambiguities. *Biosystems* 30: 161-171.

Brauch, H., Weirich, G., Brieger, J., et al., 2000, VHL alterations in human clear cell renal cell carcinoma: association with advanced tumor stage and a novel hot spot mutation. *Cancer Res* 60: 1942-1948.

Breuer, S., Gerlach, H., Kolaric, B., Urbanke, C., Opitz, N., and Geyer, M., 2006, Biochemical indication for myristoylation-dependent conformational changes in HIV-1 Nef. *Biochemistry (Mosc).* 45: 2339-2349.

Bridger, W.A., and Henderson, J.F., 1983, *Cell ATP.* Wiley, New York.

Brown, C.M., Stockwell, P.A., Dalphin, M.E., and Tate, W.P., 1994, The translational termination signal database (TransTerm) now also includes initiation contexts. *Nucleic Acids Res* 22: 3620-3624.

Brown, J.R., 2003, Ancient horizontal gene transfer. *Nat Rev Genet* 4: 121-132.

Bulmer, M., 1987, Coevolution of codon usage and transfer RNA abundance. *Nature* 325: 728-730.

Bulmer, M., 1991, The selection-mutation-drift theory of synonymous codon usage. *Genetics* 129: 897-907.

Bumann, D., Aksu, S., Wendland, M., Janek, K., Zimny-Arndt, U., Sabarth, N., Meyer, T.F., and Jungblut, P.R., 2002, Proteome analysis of secreted proteins of the gastric pathogen Helicobacter pylori. *Infect. Immun.* 70: 3396-3403.

Burge, C., and Karlin, S., 1997, Prediction of complete gene structures in human genomic dna. *J. Mol. Biol.* 268: 78-94.

Burge, C.B., 1998, Modeling dependencies in pre-mRNA splicing signals. In *Computational Methods in Molecular Biology*. Salzberg, S., Searls, D. and Kasif, S. (eds). Elsevier Science, Amsterdam, pp. 127-163.

Burge, C.B., and Karlin, S., 1998, Finding the genes in genomic DNA. *Curr Opin Struct Biol* 8: 346-354.

Burges, C.J.C., 1998, A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery* 2: 121-167.

Bury-Mone, S., Skouloubris, S., Labigne, A., and De Reuse, H., 2001, The *Helicobacter pylori* UreI protein: role in adaptation to acidity and identification of residues essential for its activity and for acid activation. *Mol. Microbiol.* 42: 1021-1034.

Carroll, J., Fearnley, I.M., Shannon, R.J., Hirst, J., and Walker, J.E., 2003, Analysis of the subunit composition of complex I from bovine heart mitochondria. *Mol Cell Proteomics* 2: 117-126.

Chang, J.T., 1996, Full reconstruction of Markov models on evolutionary trees: identifiability and consistency. *Math Biosci* 137: 51-73.

Chen, G., Gharib, T.G., Huang, C.C., et al., 2002, Discordant protein and mRNA expression in lung adenocarcinomas. *Mol Cell Proteomics* 1: 304-313.

Chen, J.J., Peck, K., Hong, T.M., et al., 2001, Global analysis of gene expression in invasion by a lung cancer model. *Cancer Res* 61: 5223-5230.

Chen, R., Pan, S., Brentnall, T.A., and Aebersold, R., 2005, Proteomic Profiling of Pancreatic Cancer for Biomarker Discovery. *Mol Cell Proteomics* 4: 523-533.

Chen, Y., Bittner, M.L., and Rougherty, E.R., 1999, Issues associated with microarray data analysis and integration. *Nat. Genet.* 22: 213-215.

Chilingaryan, A., Gevorgyan, N., Vardanyan, A., Jones, D., and Szabo, A., 2002, Multivariate approach for selecting sets of differentially expressed genes. *Math Biosci* 176: 59-69.

Cho, R.J., Campbell, M.J., Winzeler, E.A., et al., 1998, A genome-wide transcriptional analysis of the mitotic cell cycle. *Mol Cell* 2: 65-73.

Chou, P.Y., and Fasman, G.D., 1978a, Prediction of the secondary structure of proteins from their amino acid sequence. *Adv Enzymol Relat Areas Mol Biol* 47: 45-148.

Chou, P.Y., and Fasman, G.D., 1978b, Empirical predictions of protein conformation. *Annu Rev Biochem* 47: 251-276.

Chuang, S.E., Daniels, D.L., and Blattner, F.R., 1993, Global regulation of gene expression in Escherichia coli. *J Bacteriol* 175: 2026-2036.

Cigan, A.M., and Donahue, T.F., 1987, Sequence and structural features associated with translational initiator regions in yeast--a review. *Gene* 59: 1-18.

Clayton, D.A., 1982, Replication of animal mitochondrial DNA. *Cell* 28: 693-705.

Clayton, D.A., 2000, Transcription and replication of mitochondrial DNA. *Hum Reprod* 15: 11-17.

Coessens, B., Thijs, G., Aerts, S., et al., 2003, INCLUSive: A web portal and service registry for microarray and regulatory sequence analysis. *Nucleic Acids Res* 31: 3468-3470.

Coghlan, A., and Wolfe, K.H., 2000, Relationship of codon bias to mRNA concentration and protein length in Saccharomyces cerevisiae. *Yeast* 16: 1131-1145.

Colby, C., and Edlin, G., 1970, Nucleotide pool levels in growing, inhibited, and transformed chick fibroblast cells. *Biochemistry (Mosc).* 9: 917.

Comeron, J.M., and Aguade, M., 1998, An evaluation of measures of synonymous codon usage bias. *J Mol Evol* 47: 268-274.

Correa, P., 1997, *Helicobacter pylori* as a pathogen and carcinogen. *J. Physiol. Pharmacol.* 48: 19-24.

Covell, D.G., Wallqvist, A., Rabow, A.A., and Thanki, N., 2003, Molecular classification of cancer: unsupervised self-organizing map analysis of gene expression microarray data. *Mol Cancer Ther* 2: 317-332.

Crick, F.H., 1965, Recent research in molecular biology: introduction. *Br Med Bull* 21: 183-186.

Dalphin, M.E., Brown, C.M., Stockwell, P.A., and Tate, W.P., 1996, TransTerm: a database of translational signals. *Nucleic Acids Res* 24: 216-218.

Danchin, A., 2002, *The Delphic boat: what genomes tell us.* Harvard University Press, Cambridge, MA.

Darwin, C., 1859, *The Orgin of Species by means of natural selection, or the preservation of favoured races in the struggle for life.* JOHN MURRAY, London.

Davila, A.M., Lorenzini, D.M., Mendes, P.N., et al., 2005, GARSA: genomic analysis resources for sequence annotation. *Bioinformatics* 21: 4302-4303.

Dayhoff, M.O., Schwartz, R.M., and Orcutt, B.C., 1978, A model of evolutionary change in proteins. In *Atlas of Protein Sequence and Structure*. Vol. 5, Suppl. 3. Dayhoff, M.O. (ed). National Biomedical Research Foundation, Washington D.C., pp. 345-352.

de Vries, J.S., Andriotis, V.M., Wu, A.J., and Rathjen, J.P., 2006, Tomato Pto encodes a functional N-myristoylation motif that is required for signal transduction in Nicotiana benthamiana. *Plant J* 45: 31-45.

Desper, R., and Gascuel, O., 2002, Fast and accurate phylogeny reconstruction algorithms based on the minimum-evolution principle. *J. Comput. Biol.* 9: 687-705.

Diehn, M., Eisen, M.B., Botstein, D., and Brown, P.O., 2000, Large-scale identification of secreted and membrane-associated gene products using DNA microarrays. *Nat. Genet.* 25: 58-62.

Dinel, S., Bolduc, C., Belleau, P., et al., 2005, Reproducibility, bioinformatic analysis and power of the SAGE method to evaluate changes in transcriptome. *Nucleic Acids Res* 33: e26.

Dobzhansky, T., 1973, Nothing in biology makes sense except in the light of evolution. *The American Biology Teacher* 35: 125-129.

Doolittle, R.F., Hunkapiller, M.W., Hood, L.E., Devare, S.G., Robbins, K.C., Aaronson, S.A., and Antoniades, H.N., 1983, Simian sarcoma virus onc gene, v-sis, is derived from the gene (or genes) encoding a platelet-derived growth factor. *Science* 221: 275-277.

dos Reis, M., Wernisch, L., and Savva, R., 2003, Unexpected correlations between gene expression and codon usage bias from microarray data for the whole Escherichia coli K-12 genome. *Nucleic Acids Res* 31: 6976-6985.

Drake, J.W., Charlesworth, B., Charlesworth, D., and Crow, J.F., 1998, Rates of spontaneous mutation. *Genetics* 148: 1667-1686.

Drouin, G., and de Sa, M.M., 1995, The concerted evolution of 5S ribosomal genes linked to the repeat units of other multigene families. *Mol Biol Evol* 12: 481-493.

Drouin, G., Prat, F., Ell, M., and Clarke, G.D., 1999, Detecting and characterizing gene conversions between multigene family members. *Mol Biol Evol* 16: 1369-1390.

Drouin, G., 2002a, Characterization of the gene conversions between the multigene family members of the yeast genome. *J Mol Evol* 55: 14-23.

Drouin, G., 2002b, Testing claims of gene conversion between multigene family members: examples from echinoderm actin genes. *J Mol Evol* 54: 138-139.

Dunham, I., Shimizu, N., Roe, B.A., et al., 1999, The DNA sequence of human chromosome 22. *Nature* 402: 489-495.

Durbin, R., 1998, *Biological sequence analysis: probabilistic models of proteins and nucleic acids.* Cambridge University Press, Cambridge.

Duret, L., 2000, tRNA gene number and codon usage in the C. elegans genome are co-adapted for optimal translation of highly expressed genes. *Trends Genet* 16: 287-289.

Eddy, S.R., 2005, "Antedisciplinary" science. *PLoS Comput Biol* 1: e6.

Einstein, A., Russell, B., Dewey, J., et al., 1931, *Living Philosophies.* Simon and Schuster., New York.

Eisen, J.A., 2000, Horizontal gene transfer among microbial genomes: new insights from complete genome analysis. *Curr Opin Genet Dev* 10: 606-611.

Eisen, M.B., Spellman, P.T., Brown, P.O., and Botstein, D., 1998, Cluster analysis and display of genome-wide expression patterns. *Proc. Natl. Acad. Sci. USA.* 95: 14863-14868.

Engel, E., Peskoff, A., Kauffman, G.L., Jr., and Grossman, M.I., 1984, Analysis of hydrogen ion concentration in the gastric gel mucus layer. *Am. J. Physiol.* 247: G321-338.

Epstein, C.B., and Butow, R.A., 2000, Microarray technology - enhanced versatility, persistent challenge. *Curr. Opin. Biotechnol.* 11: 36-41.

Evans, T., Felsenfeld, G., and Reitman, M., 1990, Control of globin gene transcription. *Annu Rev Cell Biol* 6: 95-124.

Eyre-Walker, A., 1996, Synonymous codon bias is related to gene length in Escherichia coli: selection for translational accuracy? *Mol Biol Evol* 13: 864-872.

Ezzell, C., 2002, Proteins rule. *Sci. Am.* 286: 40-47.

Farazi, T.A., Waksman, G., and Gordon, J.I., 2001, The Biology and Enzymology of Protein N-Myristoylation. *J. Biol. Chem.* 276: 39501-39504.

Fasman, G.D., and Chou, P.Y., 1974, Prediction of protein conformation: consequences and aspirations. In *Peptides, polypeptides and proteins.* Blout, E.R., Bovey, F.A., Goodman, M. and Latan, N. (eds). Wiley, New York, pp. 114-125.

Fatemi, M., Hermann, A., Pradhan, S., and Jeltsch, A., 2001, The activity of the murine DNA methyltransferase Dnmt1 is controlled by interaction of the catalytic domain with the N-terminal part of the enzyme leading to an allosteric activation of the enzyme after binding to methylated DNA. *J. Mol. Biol.* 309: 1189-1199.

Felsenstein, J., 1978a, The number of evolutionary trees. *Syst. Zool.* 27: 27-33.

Felsenstein, J., 1978b, Cases in which parsimony and compatibility methods will be positively misleading. *Syst. Zool.* 27: 401-410.

Felsenstein, J., 1981, Evolutionary trees from DNA sequences: a maximum likelihood approach. *J. Mol. Evol.* 17: 368-376.

Felsenstein, J., 1988, Phylogenies from molecular sequences: inference and reliability. *Annu Rev Genet* 22: 521-565.

Felsenstein, J., and Churchill, G.A., 1996, A Hidden Markov Model approach to variation among sites in rate of evolution. *Mol Biol Evol* 13: 93-104.

Felsenstein, J., 2002, PHYLIP 3.6 (phylogeny inference package). Seattle: Department of Genetics, University of Washington.

Felsenstein, J., 2004, *Inferring phylogenies.* Sinauer, Sunderland, Massachusetts.

Figeys, D., 2002, Adapting arrays and lab-on-a-chip technology for proteomics. *Proteomics* 2: 373-382.

Figeys, D., 2003a, Proteomics in 2002: a year of technical development and wide-ranging applications. *Anal Chem* 75: 2891-2905.

Figeys, D., 2003b, Novel approaches to map protein interactions. *Curr Opin Biotechnol* 14: 119-125.

Fisher, R.A., 1926, The arrangement of field experiments. *Journal of the Ministry of Agriculture* 33.

Fisher, R.A., 1936, The use of multiple measurements in taxonomic problems. *Annals of Eugenics* 7: 179-188.

Fitch, W.M., and Margoliash, E., 1967, Construction of phylogenetic trees. *Science* 155: 279-284.

Fitch, W.M., 1971, Toward defining the course of evolution: minimum change for a specific tree topology. *Syst. Zool.* 20: 406-416.

Fleischmann, R.D., Adams, M.D., White, O., et al., 1995, Whole-genome random sequencing and assembly of Haemophilus influenzae Rd. *Science* 269: 496-512.

Foissac, S., and Schiex, T., 2005, Integrating alternative splicing detection into gene prediction. *BMC Bioinformatics* 6: 25.

Folley, L.S., and Fox, T.D., 1991, Site-directed mutagenesis of a Saccharomyces cerevisiae mitochondrial translation initiation codon. *Genetics* 129: 659-668.

Fong, T.C., and Emerson, B.M., 1992, The erythroid-specific protein cGATA-1 mediates distal enhancer activity through a specialized beta-globin TATA box. *Genes Dev* 6: 521-532.

Forde, C.E., and McCutchen-Maloney, S.L., 2002, Characterization of transcription factors by mass spectrometry and the role of SELDI-MS. *Mass Spectrom Rev* 21: 419-439.

Francino, M.P., and Ochman, H., 1997, Strand asymmetries in DNA evolution. *Trends Genet.* 13: 240-245.

Fraser, C.M., Gocayne, J.D., White, O., et al., 1995, The minimal gene complement of Mycoplasma genitalium. *Science* 270: 397-403.

Frederico, L.A., Kunkel, T.A., and Shaw, B.R., 1990, A sensitive genetic assay for the detection of cytosine deamination: determination of rate constants and the activation energy. *Biochemistry (Mosc).* 29: 2532-2537.

Frederico, L.A., Kunkel, T.A., and Shaw, B.R., 1993, Cytosine deamination in mismatched base pairs. *Biochemistry (Mosc).* 32: 6523-6530.

Freeman, J.M., Plasterer, T.N., Smith, T.F., and Mohr, S.C., 1998, Patterns of Genome Organization in Bacteria. *Science* 279: 1827a-.

Freund, Y.a.S., R. E., 1998, Large margin classification using the perceptron algorithm. In *The 11th Annual Conference on Computational Learning Theory (COLT' 98).* ACM Press.

Frishman, D., Mironov, A., Mewes, H.W., and Gelfand, M., 1998, Combining diverse evidence for gene recognition in completely sequenced bacterial genomes. *Nucleic Acids Res* 26: 2941-2947.

Futcher, B., Latter, G.I., Monardo, P., McLaughlin, C.S., and Garrels, J.I., 1999, A sampling of the yeast proteome. *Mol Cell Biol* 19: 7357-7368.

Gaasterland, T., and Bekiranov, S., 2000, Making the most of microarray data [news]. *Nat. Genet.* 24: 204-206.

Galtier, N., and Lobry, J.R., 1997, Relationships between genomic G+C content, RNA secondary structures, and optimal growth temperature in prokaryotes. *J Mol Evol* 44: 632-636.

Gelfand, M.S., 1989, Statistical analysis of mammalian pre-mRNA splicing sites. *Nucleic Acids Res* 17: 6369-6382.

Gelfand, M.S., 1992, Statistical analysis and prediction of the exonic structure of human genes. *J Mol Evol* 35: 239-252.

Gelfand, M.S., Kozhukhin, C.G., and Pevzner, P.A., 1992, Extendable words in nucleotide sequences. *Comput Appl Biosci* 8: 129-135.

Geman, S., and Geman, D., 1984, Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 6: 721-741.

Gibbs, A., and Primrose, S., 1976, A correlation between the genome compositions of bacteriophages and their hosts. *Intervirology* 7: 351-355.

Gissi, C., Iannelli, F., and Pesole, G., 2004, Complete mtDNA of Ciona intestinalis reveals extensive gene rearrangement and the presence of an atp8 and an extra trnM gene in ascidians. *J Mol Evol* 58: 376-389.

Goffeau, A., and al., e., 1996, Life with 6000 genes. *Science* 274: 546.

Gojobori, T., Li, W.-H., and Graur, D., 1982, Patterns of nucleotide substitution in pseudogenes and functional genes. *J. Mol. Evol.* 18: 360-369.

Golding, G.B., 1983, Estimates of DNA and protein sequence divergence: An examination of some assumptions. *Mol. Biol. Evol.* 1: 125-142.

Goldman, N., and Yang, Z., 1994, A codon-based model of nucleotide substitution for protein-coding DNA sequences. *Mol. Biol. Evol.* 11: 725-736.

Gora-Sochacka, A., 2004, Viroids: unusual small pathogenic RNAs. *Acta Biochim Pol* 51: 587-607.

Gordon, D., Abajian, C., and Green, P., 1998, Consed: a graphical tool for sequence finishing. *Genome Res.* 8: 195-202.

Gouy, M., and Gautier, C., 1982, Codon usage in bacteria: correlation with gene expressivity. *Nucleic Acids Res.* 10: 7055-7064.

Graveley, B.R., 2005, Mutually exclusive splicing of the insect Dscam pre-mRNA directed by competing intronic RNA secondary structures. *Cell* 123: 65-73.

Griffin, T.J., Gygi, S.P., Ideker, T., Rist, B., Eng, J., Hood, L., and Aebersold, R., 2002, Complementary profiling of gene expression at the transcriptome and proteome levels in Saccharomyces cerevisiae. *Mol Cell Proteomics* 1: 323-333.

Grigoriev, A., 1998, Analyzing genomes with cumulative skew diagrams. *Nucleic Acids Res* 26: 2286-2290.

Grunert, S., and Jackson, R.J., 1994, The immediate downstream codon strongly influences the efficiency of utilization of eukaryotic translation initiation codons. *Embo J* 13: 3618-3630.

Gu, X., Hewett-Emmett, D., and Li, W.H., 1998, Directional mutational pressure affects the amino acid composition and hydrophobicity of proteins in bacteria. *Genetica* 102-103: 383-391.

Gumbel, E.J., 1958, *Statistics of extremes.* Columbia University Press, New York, NY.

Gusfield, D., 1997, *Algorithms on strings, trees, and sequences: computer science and computational biology.* Cambridge University Press, Cambridge.

Gygi, S.P., Rochon, Y., Franza, B.R., and Aebersold, R., 1999, Correlation between protein and mRNA abundance in yeast. *Mol Cell Biol* 19: 1720-1730.

Haldane, J.B.S., 1937, The effect of variation of fitness. *Amer. Nat.* 71: 337-349.

Hamajima, N., Goto, Y., Nishio, K., Tanaka, D., Kawai, S., Sakakibara, H., and Kondo, T., 2004, *Helicobacter pylori* eradication as a preventive tool against gastric cancer. *Asian Pacific Journal of Cancer Prevention* 5: 246-252.

Harkins, S., Cornell, C.T., and Whitton, J.L., 2005, Analysis of translational initiation in coxsackievirus B3 suggests an alternative explanation for the high frequency of R+4 in the eukaryotic consensus motif. *J Virol* 79: 987-996.

Hartigan, J.A., 1975, *Clustering algorithms.* Wiley, New York.

Hastings, W.K., 1970, Monte Carlo sampling methods using Markov chain and their applications. *Biometrika* 57: 97-109.

Hayes, W.S., and Borodovsky, M., 1998, How to interpret an anonymous bacterial genome: machine learning approach to gene identification. *Genome Res.* 8: 1154-1171.

Heckman, J.E., Sarnoff, J., Alzner-DeWeerd, B., Yin, S., and RajBhandary, U.L., 1980, Novel features in the genetic code and codon reading patterns in Neurospora crassa mitochondria based on sequences of six mitochondrial tRNAs. *Proc. Natl. Acad. Sci. U S A* 77: 3159-3163.

Hein, J., 1990, A unified approach to phylogenies and alignments. *Methods in Enzymology* 183: 625-644.

Hein, J., 1994, TreeAlign. *Methods Mol Biol* 25: 349-364.

Hendy, M.D., and Penny., D., 1982, Branch and bound algorithms to determine minimal evolutionary trees. *Math. Biosci.* 60: 133-142.

Hendy, M.D., and Penny, D., 1989, A framework for the quantitative study of evolutionary trees. *Syst. Zool.* 38: 297-309.

Henikoff, S., and Henikoff, J.G., 1992, Amino acid substitution matrices from protein blocks. *Proc. Natl. Acad. Sci. U S A* 89: 10915-10919.

Heyer, L.J., Kruglyak, S., and Yooseph, S., 1999, Exploring expression data: identification and analysis of coexpressed genes. *Genome Res.* 9: 1106-1115.

Hickey, D.A., Bally-Cuif, L., Abukashawa, S., Payant, V., and Benkel, B.F., 1991, Concerted evolution of duplicated protein-coding genes in Drosophila. *Proc. Natl. Acad. Sci. U S A* 88: 1611-1615.

Hickey, D.A., and Singer, G.A., 2004, Genomic and proteomic adaptations to growth at high temperature. *Genome Biol* 5: 117.

Hickson, R.E., Simon, C., and Perrey, S.W., 2000, The performance of several multiple-sequence alignment programs in relation to secondary-structure features for an rRNA sequence. *Mol Biol Evol* 17: 530-539.

Higgins, D.G., and Sharp, P.M., 1988, CLUSTAL: a package for performing multiple sequence alignment on a microcomputer. *Gene* 73: 237-244.

Higgs, P.G., and Attwood, T.K., 2004, *Bioinformatics and molecular evolution.* Blackwell, Malden.

Hillis, D.M., Moritz, C., and Mable, B.K., 1996, *Molecular systematics.* Sinauer Associates, Inc., Sunderland, Massachusetts.

Hirst, J.D., and Sternberg, M.J., 1991, Prediction of ATP/GTP-binding motif: a comparison of a perceptron type neural network and a consensus sequence method [corrected]. *Protein Eng* 4: 615-623.

Hodges, P.E., McKee, A.H., Davis, B.P., Payne, W.E., and Garrels, J.I., 1999, The Yeast Proteome Database (YPD): a model for the organization and presentation of genome-wide functional data. *Nucleic Acids Res* 27: 69-73.

Hofer, A., Steverding, D., Chabes, A., Brun, R., and Thelander, L., 2001, *Trypanosoma brucei* CTP synthetase: a target for the treatment of African sleeping sickness. *Proc. Natl. Acad. Sci. U S A* 98: 6412-6416.

Hoffmann, R.J., Boore, J.L., and Brown, W.M., 1992, A novel mitochondrial genome organization for the blue mussel, Mytilus edulis. *Genetics* 131: 397-412.

Holmes, I., and Bruno, W.J., 2001, Evolutionary HMMs: a Bayesian approach to multiple alignment. *Bioinformatics* 17: 803-820.

Holstege, F.C., Jennings, E.G., Wyrick, J.J., Lee, T.I., Hengartner, C.J., Green, M.R., Golub, T.R., Lander, E.S., and Young, R.A., 1998, Dissecting the regulatory circuitry of a eukaryotic genome. *Cell* 95: 717-728.

Holt, I.J., Lorimer, H.E., and Jacobs, H.T., 2000, Coupled leading- and lagging-strand synthesis of mammalian mitochondrial DNA. *Cell* 100: 515-524.

Holt, I.J., and Jacobs, H.T., 2003, Response: The mitochondrial DNA replication bubble has not burst. *Trends Biochem Sci* 28: 355-356.

Hopf, F.A., Michod, R.E., and Sanderson, M.J., 1988, The effect of the reproductive system on mutation load. *Theor Popul Biol* 33: 243-265.

Huang, X., 1993, Alignment of Three Sequences in Quadratic Space. *Applied Computing Review* 1: 7-11.

Huang, X., Wang, J., Aluru, S., Yang, S.P., and Hillier, L., 2003, PCAP: a whole-genome assembly program. *Genome Res.* 13: 2164-2170.

Huang, X.Q., 1992, A Contig Assembly Program Based on Sensitive Detection of Fragment Overlaps. *Genomics* 14: 18-25.

Huelsenbeck, J.P., Ronquist, F., Nielsen, R., and Bollback, J.P., 2001, Bayesian inference of phylogeny and its impact on evolutionary biology. *Science* 294: 2310-2314.

Huelsenbeck, J.P., Larget, B., and Alfaro, M.E., 2004, Bayesian phylogenetic model selection using reversible jump Markov chain Monte Carlo. *Mol Biol Evol* 21: 1123-1133.

Hughes, A.L., da Silva, J., and Friedman, R., 2001, Ancient genome duplications did not structure the human Hox-bearing chromosomes. *Genome Res.* 11: 771-780.

Hunt, R.H., 2004, Will eradication of *Helicobacter pylori* infection influence the risk of gastric cancer? *Am. J. Med.* 117: 86S-91S.

Hurst, L.D., and Merchant, A.R., 2001, High guanine-cytosine content is not an adaptation to high temperature: a comparative analysis amongst prokaryotes. *Proc. R. Soc. Lond. B.* 268: 493-497.

Huynen, M., Dandekar, T., and Bork, P., 1998, Differential genome analysis applied to the species-specific features of *Helicobacter pylori*. *FEBS Lett.* 426: 1-5.

Ideker, T., Thorsson, V., Ranish, J.A., et al., 2001, Integrated genomic and proteomic analyses of a systematically perturbed metabolic network. *Science* 292: 929-934.

Ikemura, T., 1981, Correlation between the abundance of *Escherichia coli* transfer RNAs and the occurrence of the respective codons in its protein genes: a proposal for a synonymous codon choice that is optimal for the E coli translational system. *J Mol Biol* 151: 389-409.

Ikemura, T., 1982, Correlation between the abundance of yeast transfer RNAs and the occurrence of the respective codons in protein genes. Differences in synonymous codon choice patterns of yeast and Escherichia coli with reference to the abundance of isoaccepting transfer RNAs. *J Mol Biol* 158: 573-597.

Ikemura, T., 1985, Codon usage and tRNA content in unicellular and multicellular organisms. *Mol Biol Evol* 2: 13-34.

Ikemura, T., 1992, Correlation between codon usage and tRNA content in microorganisms. In *Transfer RNA in protein synthesis.* Hatfield, D.L., Lee, B. and Pirtle, J. (eds). CRC Press, Boca Raton, Fla., pp. 87-111.

Inagaki, Y., Kojima, A., Bessho, Y., Hori, H., Ohama, T., and Osawa, S., 1995, Translation of synonymous codons in family boxes by

Mycoplasma capricolum tRNAs with unmodified uridine or adenosine at the first anticodon position. *J Mol Biol* 251: 486-492.

Istrail, S., Sutton, G.G., Florea, L., et al., 2004, Whole-genome shotgun assembly and comparison of human genome assemblies. *Proc. Natl. Acad. Sci. U S A* 101: 1916-1921. Epub 2004 Feb 1909.

Jacob, F., 1988, *The statue within: an autobiography.* Basic Books, Inc., New York.

Jayaswal, V., Jermiin, L.S., and Robinson, J., 2005, Estimation of Phylogeny Using a General Markov Model. *Evolutionary Bioinformatics Online* 1: 62-80.

Jensen, J.L., and Hein, J., 2005, Gibbs sampler for statistical multiple alignment. *Statistica Sinica* 15: 889-907.

Jermiin, L., Graur, D., and Crozier, R., 1995, Evidence from Analyses of Intergenic Regions for Strand-specific Directional Mutation Pressure in Metazoan Mitochondrial DNA. *Mol Biol Evol* 12: 558-563.

Jia, M., and Li, Y., 2005, The relationship among gene expression, folding free energy and codon usage bias in Escherichia coli. *FEBS Lett* 579: 5333-5337.

Jin, L., and Nei, M., 1990, Limitations of the evolutionary parsimony method of phylogenetic analysis. *Mol. Biol. Evol.* 7: 82-102.

Jones, D.T., Taylor, W.R., and Thornton, J.M., 1992, The rapid generation of mutation data matrices from protein sequences. *Comput Appl Biosci* 8: 275-282.

Jukes, T.H., and Cantor, C.R., 1969, Evolution of protein molecules. In *Mammalian protein metabolism.* Munro, H.N. (ed). Academic Press, New York, pp. 21-123.

Kanaya, S., Yamada, Y., Kudo, Y., and Ikemura, T., 1999, Studies of codon usage and tRNA genes of 18 unicellular organisms and quantification of Bacillus subtilis tRNAs: gene expression level and species-specific diversity of codon usage based on multivariate analysis. *Gene* 238: 143-155.

Kaneko, T., Tanaka, A., Sato, S., Kotani, H., Sazuka, T., Miyajima, N., Sugiura, M., and Tabata, S., 1995, Sequence analysis of the genome of the unicellular cyanobacterium Synechocystis sp. strain PCC6803. I. Sequence features in the 1 Mb region from map positions 64% to 92% of the genome. *DNA Res* 2: 153-166, 191-158.

Kaneko, T., Sato, S., Kotani, H., et al., 1996, Sequence analysis of the genome of the unicellular cyanobacterium Synechocystis sp. strain PCC6803. II. Sequence determination of the entire genome and assignment of potential protein-coding regions. *DNA Res* 3: 109-136.

Katinka, M.D., Duprat, S., Cornillot, E., et al., 2001, Genome sequence and gene compaction of the eukaryote parasite Encephalitozoon cuniculi. *Nature* 414: 450-453.

Kazan, K., 2003, Alternative splicing and proteome diversity in plants: the tip of the iceberg has just emerged. *Trends Plant Sci* 8: 468-471.

Kent, W.J., and Haussler, D., 2001, Assembly of the working draft of the human genome with GigAssembler. *Genome Res.* 11: 1541-1548.

Kim, D.W., Lee, K.H., and Lee, D., 2005, Detecting clusters of different geometrical shapes in microarray gene expression data. *Bioinformatics* 21: 1927-1934.

Kimura, M., and Ohta, T., 1972, On the stochastic model for estimation of mutational distance between homologous proteins. *J. Mol. Evol.* 2: 87-90.

Kimura, M., 1980, A simple method for estimating evolutionary rates of base substitutions through comparative studies of nucleotide sequences. *J. Mol. Evol.* 16: 111-120.

Kishino, H., Miyata, T., and Hasegawa, M., 1990, Maximum likelihood inference of protein phylogeny and the origin of chloroplasts. *J. Mol. Evol.* 31: 151-160.

Kjer, K.M., 1995, Use of Ribosomal-RNA Secondary Structure in Phylogenetic Studies to Identify Homologous Positions - an Example of Alignment and Data Presentation from the Frogs. *Mol. Phylogenet. Evol.* 4: 314-330.

Kohonen, T., 2001, *Self-Organizing Maps.* Springer, Berlin.

Kolaczkowski, B., and Thornton, J.W., 2004, Performance of maximum parsimony and likelihood phylogenetics when evolution is heterogeneous. *Nature* 431: 980-984.

Kolkman, A., Dirksen, E.H., Slijper, M., and Heck, A.J., 2005, Double standards in quantitative proteomics: direct comparative assessment of difference in gel electrophoresis and metabolic stable isotope labeling. *Mol Cell Proteomics* 4: 255-266.

Kondow, A., Yokobori, S., Ueda, T., and Watanabe, K., 1998, Ascidian mitochondrial tRNA(Met) possessing unique structural characteristics. *Nucleosides Nucleotides* 17: 531-539.

Kondrashov, A.S., and Crow, J.F., 1988, King's formula for the mutation load with epistasis. *Genetics* 120: 853-856.

Koonin, E.V., 2003, Horizontal gene transfer: the path to maturity. *Mol. Microbiol.* 50: 725-727.

Kornberg, A., and Baker, T.A., 1992, *DNA replication.* Freeman, New York.

Kornblihtt, A.R., 2005, Promoter usage and alternative splicing. *Curr Opin Cell Biol* 17: 262-268.

Koski, L.B., Gray, M.W., Lang, B.F., and Burger, G., 2005, AutoFACT: an automatic functional annotation and classification tool. *BMC Bioinformatics* 6: 151.

Kozak, M., 1982, How do eukaryotic ribosomes recognize the unique AUG initiator codon in messenger RNA? *Biochem Soc Symp* 47: 113-128.

Kozak, M., 1984, Selection of initiation sites by eucaryotic ribosomes: effect of inserting AUG triplets upstream from the coding sequence for preproinsulin. *Nucleic Acids Res* 12: 3873-3893.

Kozak, M., 1989, The scanning model for translation: an update. *J Cell Biol* 108: 229-241.

Krogh, A., Mian, I.S., and Haussler, D., 1994, A hidden Markov model that finds genes in E. coli DNA. *Nucleic Acids Res* 22: 4768-4778.

Kumar, S., Tamura, K., Jakobsen, I.B., and Nei, M., 2001, MEGA2: molecular evolutionary genetics analysis software. *Bioinformatics* 17: 1244-1245.

Kurland, C.G., Canback, B., and Berg, O.G., 2003, Horizontal gene transfer: a critical view. *Proc. Natl. Acad. Sci. U S A* 100: 9658-9662.

La Scola, B., Audic, S., Robert, C., Jungang, L., de Lamballerie, X., Drancourt, M., Birtles, R., Claverie, J.M., and Raoult, D., 2003, A giant virus in amoebae. *Science* 299: 2033.

Laemmli, U.K., 1970, Cleavage of structural proteins during the assembly of the head of bacteriophage T4. *Nat. Biotechnol.* 227: 680-685.

Lake, J.A., 1994, Reconstructing evolutionary trees from DNA and protein sequences: paralinear distances. *Proc. Natl. Acad. Sci. USA.* 91: 1455-1459.

Lamendola, D.E., Duan, Z., Yusuf, R.Z., and Seiden, M.V., 2003, Molecular description of evolving paclitaxel resistance in the SKOV-3 human ovarian carcinoma cell line. *Cancer Res* 63: 2200-2205.

Lander, E.S., and Linton, L.M., and Birren, B., et al., 2001a, Initial sequencing and analysis of the human genome. *Nature* 409: 860-921.

Lander, E.S., and Linton, L.M., and Birren, B., et al., 2001b, Initial sequencing and analysis of the human genome. [see comments]. [erratum appears in Nature 2001 Jun 7;411(6838):720]. *Nature* 409: 860-921.

Lang, B.F., Burger, G., J., O.C., Cedergren, R., Golding, G.B., Lemieux, C., Sankoff, D., Turmel, M., and Gray, M.W., 1997, An ancestral mitochondrial DNA resembling a eubacterial genome in miniature [see comments]. *Nature* 387: 493-497.

Lawrence, C.E., Altschul, S.F., Boguski, M.S., Liu, J.S., Neuwald, A.F., and Wootton, J.C., 1993, Detecting subtle sequence signals: a Gibbs sampling strategy for multiple alignment. *Science* 262: 208-214.

Lee, C., and Wang, Q., 2005, Bioinformatics analysis of alternative splicing. *Brief Bioinform* 6: 23-33.

Lerner, B., Guterman, H., Dinstein, I., and Romem, Y., 1995, Human chromosome classification using multilayer perceptron neural network. *Int J Neural Syst* 6: 359-370.

Lesk, A.M., 2004, *Introduction to protein science: Architecture, function and genomics.* Oxford University Press, New York.

Letunic, I., Goodstadt, L., Dickens, N.J., et al., 2002, Recent improvements to the SMART domain-based sequence annotation resource. *Nucleic Acids Res* 30: 242-244.

Letunic, I., Copley, R.R., Schmidt, S., Ciccarelli, F.D., Doerks, T., Schultz, J., Ponting, C.P., and Bork, P., 2004, SMART 4.0: towards genomic data integration. *Nucleic Acids Res* 32: D142-144.

Li, C.C., 1976, *First course in population genetics.* The Boxwood Press, Pacific Grove, California.

Li, W.-H., 1997, *Molecular evolution.* Sinauer, Sunderland, Massachusetts.

Li, W.H., Wu, C.I., and Luo, C.C., 1984, Nonrandomness of point mutation as reflected in nucleotide substitutions in pseudogenes and its evolutionary implications. *J. Mol. Evol.* 21: 58-71.

Liebler, D.C., III., T.b.D.C.L.f.b.J.R.Y., and Publisher, c., 2002, *Introduction to proteomics: tools for the new biology.* Humana Press, Totowa, NJ.

Liljenstrom, H., and von Heijne, G., 1987, Translation rate modification by preferential codon usage: intragenic position effects. *J Theor Biol* 124: 43-55.

Lindahl, T., 1993, Instability and decay of the primary structure of DNA. *Nature* 362: 709-715.

Lipman, D.J., and Pearson, W.R., 1985, Rapid and sensitive protein similarity searches. *Science* 227: 1435-1441.

Lipscombe, D., 2005, Neuronal proteins custom designed by alternative splicing. *Curr Opin Neurobiol* 15: 358-363.

Lithwick, G., and Margalit, H., 2005, Relative predicted protein levels of functionally associated proteins are conserved across organisms. *Nucleic Acids Res* 33: 1051-1057.

Lobry, J.R., 1996, Asymmetric substitution patterns in the two DNA strands of bacteria. *Mol Biol Evol* 13: 660-665.

Lobry, J.R., and Sueoka, N., 2002, Asymmetric directional mutation pressures in bacteria. *Genome Biol* 3: research58.51-14.

Lobry, J.R., 2004, Life history traits and genome structure: aerobiosis and G+C content in bacteria. *Lecture Notes in Computer Science* 3039: 679-686.

Lockhart, P.J., Steel, M.A., Hendy, M.D., and Penny, D., 1994, Recovering evolutionary trees under a more realistic model of sequence evolution. *Mol Biol Evol* 11: 605-612.

Lowe, T.M., and Eddy, S.R., 1997, tRNAscan-SE: a program for improved detection of transfer RNA genes in genomic sequence. *Nucleic Acids Res* 25: 955-964.

Lowry, J.A., and Atchley, W.R., 2000, Molecular evolution of the GATA family of transcription factors: conservation within the DNA-binding domain. *J. Mol. Evol.* 50: 103-115.

Madden, S.L., Galella, E.A., Zhu, J., Bertelsen, A.H., and Beaudry, G.A., 1997, SAGE transcript profiles for p53-dependent growth regulation. *Oncogene* 15: 1079-1085.

Mannella, C.A., Neuwald, A.F., and Lawrence, C.E., 1996, Detection of likely transmembrane beta strand regions in sequences of mitochondrial pore proteins using the Gibbs sampler. *J Bioenerg Biomembr* 28: 163-169.

Mao, C., Cushman, J.C., May, G.D., and Weller, J.W., 2003, ESTAP--an automated system for the analysis of EST data. *Bioinformatics* 19: 1720-1722.

Marcelino, L.A., Andre, P.C., Khrapko, K., Coller, H.A., Griffith, J., and Thilly, W.G., 1998, Chemically induced mutations in mitochondrial DNA of human cells: mutational spectrum of N-methyl-N'-nitro-N-nitrosoguanidine. *Cancer Res.* 58: 2857-2862.

Marchler-Bauer, A., Panchenko, A.R., Shoemaker, B.A., Thiessen, P.A., Geer, L.Y., and Bryant, S.H., 2002, CDD: a database of conserved domain alignments with links to domain three-dimensional structure. *Nucleic Acids Res* 30: 281-283.

Marchler-Bauer, A., Anderson, J.B., Cherukuri, P.F., et al., 2005, CDD: a Conserved Domain Database for protein classification. *Nucleic Acids Res* 33: D192-196.

Margulis, L., 1970, *Origin of eukaryotic cells; evidence and research implications for a theory of the origin and evolution of microbial, plant, and animal cells on the Precambrian earth.* Yale University Press, New Haven.

Martin, D.M., Berriman, M., and Barton, G.J., 2004, GOtcha: a new method for prediction of protein function assessed by the annotation of seven genomes. *BMC Bioinformatics* 5: 178.

Martin, R.P., Sibler, A.P., Gehrke, C.W., Kuo, K., Edmonds, C.G., McCloskey, J.A., and Dirheimer, G., 1990, 5-[[(carboxymethyl)amino]methyl]uridine is found in the anticodon of yeast mitochondrial tRNAs recognizing two-codon families ending in a purine. *Biochemistry (Mosc).* 29: 956-959.

Matin, A., Zychlinsky, E., Keyhan, M., and Sachs, G., 1996, Capacity of *Helicobacter pylori* to generate ionic gradients at low pH is similar to that of bacteria which grow under strongly acidic conditions. *Infect. Immun.* 64: 1434-1436.

Matsuyama, S., Ueda, T., Crain, P.F., McCloskey, J.A., and Watanabe, K., 1998, A novel wobble rule found in starfish mitochondria. Presence of 7-methylguanosine at the anticodon wobble position expands decoding capability of tRNA. *J. Biol. Chem.* 273: 3363-3368.

Maynard Smith, J., 1989, *Evolutionary genetics.* Oxford University Press, Oxford.

McCreight, E.M., 1976, A Space-Economical Suffix Tree Construction Algorithm. *Jounral of Algorithms* 23: 262-272.

McInerney, J.O., 1998, Replicational and transcriptional selection on codon usage in Borrelia burgdorferi. *Proc. Natl. Acad. Sci. U S A* 95: 10698-10703.

McLean, M.J., Wolfe, K.H., and Devine, K.M., 1998, Base composition skews, replication orientation, and gene orientation in 12 prokaryote genomes. *J. Mol. Evol.* 47: 691-696.

McLysaght, A., Hokamp, K., and Wolfe, K.H., 2002, Extensive genomic duplication during early chordate evolution. *Nat. Genet.* 31: 200-204.

Medawar, P.B., and Medawar, J.S., 1983, *Aristotle to zoos: a philosophical dictionary of biology.* Harvard University Press, Cambridge, Mass.

Medigue, C., Rouxel, T., Vigier, P., Henaut, A., and Danchin, A., 1991, Evidence for horizontal gene transfer in Escherichia coli speciation. *J Mol Biol* 222: 851-856.

Menaker, R.J., Sharaf, A.A., and Jones, N.L., 2004, *Helicobacter pylori* Infection and Gastric Cancer: Host, Bug, Environment, or All Three? *Current Gastroenterology Reports* 6: 429-435.

Mendz, G.L., and Hazell, S.L., 1996, The urea cycle of *Helicobacter pylori*. *Microbiology* 142: 2959-2967.

Metropolis, N., Rosenbluth, A.W., Rosenbluth, M.N., Teller, A.H., and Teller, E., 1953, Equation of state calculations by fast computing machines. *J. Chem. Phys.* 21: 1087-1092.

Meyer, I.M., and Durbin, R., 2004, Gene structure conservation aids similarity based gene prediction. *Nucleic Acids Res* 32: 776-783.

Mills, M., Lacroix, L., Arimondo, P.B., Leroy, J.L., Francois, J.C., Klump, H., and Mergny, J.L., 2002, Unusual DNA conformations: implications for telomeres. *Curr Med Chem Anti-Canc Agents* 2: 627-644.

Minsky, M.L., and Papert, S.A., 1969, *Perceptrons: an introduction to computational geometry.* MIT Press, Cambridge, Mass.

Mobley, H.L., Hu, L.T., and Foxal, P.A., 1991, *Helicobacter pylori* urease: properties and role in pathogenesis. *Scandinavian Journal of Gastroenterology Supplement* 187: 39-46.

Moerschell, R.P., Hosokawa, Y., Tsunasawa, S., and Sherman, F., 1990, The specificities of yeast methionine aminopeptidase and acetylation of amino-terminal methionine in vivo. Processing of altered iso-1-cytochromes c created by oligonucleotide transformation. *J. Biol. Chem.* 265: 19638-19643.

Moi, P., Loudianos, G., Lavinha, J., et al., 1992, Delta-thalassemia due to a mutation in an erythroid-specific binding protein sequence 3' to the delta-globin gene. *Blood* 79: 512-516.

Moriya, J., Yokogawa, T., Wakita, K., et al., 1994, A novel modified nucleoside found at the first position of the anticodon of methionine tRNA from bovine liver mitochondria. *Biochemistry (Mosc).* 33: 2234-2239.

Mulero, J.J., and Fox, T.D., 1994, Reduced but accurate translation from a mutant AUA initiation codon in the mitochondrial COX2 mRNA of Saccharomyces cerevisiae. *Mol Gen Genet* 242: 383-390.

Murtagh, F., 1984, Complexities of hierarchic clustering algorithms: state of the art. *Computational Statistics Quarterly* 1: 101-113.

Muse, S.V., and Gaut, B.S., 1994, A likelihood approach for comparing synonymous and nonsynonymous nucleotide substitution rates, with application to the chloroplast genome. *Mol Biol Evol* 11: 715-724.

Myers, E.W., Sutton, G.G., Delcher, A.L., et al., 2000, A whole-genome assembly of Drosophila. *Science* 287: 2196-2204.

Nakai, K., and Horton, P., 1999, PSORT: a program for detecting sorting signals in proteins and predicting their subcellular localization. *Trends Biochem Sci* 24: 34-36.

Nakashima, H., Fukuchi, S., and Nishikawa, K., 2003, Compositional changes in RNA, DNA and proteins for bacterial adaptation to higher and lower temperatures. *J Biochem (Tokyo)* 133: 507-513.

Needleman, S.B., and Wunsch, C.D., 1970, A general method applicable to the search of similarities in the amino acid sequence of two proteins. *J. Mol. Biol.* 48: 443-453.

Nei, M., and Gojobori, T., 1986, Simple methods for estimating the numbers of synonymous and nonsynonymous nucleotide substitutions. *Mol Biol Evol* 3: 418-426.

Nei, M., 1987, *Molecular Evolutionary Genetics.* Columbia University Press, New York.

Nei, M., 1996, Phylogenetic analysis in molecular evolutionary genetics. *Annu. Rev. Genet.* 30: 371-403.

Nei, M., and Kumar, S., 2000, *Molecular evolution and phylogenetics.* Oxford University Press, New York.

Nett, J.H., Kessl, J., Wenz, T., and Trumpower, B.L., 2001, The AUG start codon of the Saccharomyces cerevisiae NFS1 gene can be substituted for by UUG without increased initiation of translation at downstream codons. *Eur J Biochem* 268: 5209-5214.

Neuwald, A.F., Liu, J.S., and Lawrence, C.E., 1995, Gibbs motif sampling: detection of bacterial outer membrane protein repeats. *Protein Sci* 4: 1618-1632.

Nishimura, S., Takahashi, S., Kuroha, T., Suwabe, N., Nagasawa, T., Trainor, C., and Yamamoto, M., 2000, A GATA box in the GATA-1 gene hematopoietic enhancer is a critical element in the network of GATA factors and sites that regulate this gene. *Mol Cell Biol* 20: 713-723.

Nomenclature Committee of the International Union of Biochemistry, 1985, Nomenclature for incompletely specified bases in nucleic acid sequences. Recommendations 1984. *Eur. J. Biochem.* 150: 1-5.

Notredame, C., O'Brien, E.A., and Higgins, D.G., 1997, RAGA: RNA sequence alignment by genetic algorithm. *Nucleic Acids Res* 25: 4570-4580.

Ohno, M., Fukagawa, T., Lee, J.S., and Ikemura, T., 2002, Triplex-forming DNAs in the human interphase nucleus visualized in situ by polypurine/polypyrimidine DNA probes and antitriplex antibodies. *Chromosoma* 111: 201-213.

Ordway, J.M., Fenster, S.D., Ruan, H., and Curran, T., 2005, A transcriptome map of cellular transformation by the fos oncogene. *Mol Cancer* 4: 19.

Orkin, S.H., 1990, Globin gene regulation and switching: circa 1990. *Cell* 63: 665-672.

Orkin, S.H., 1992, GATA-binding transcription factors in hematopoietic cells. *Blood* 80: 575-581.

Page, R.D.M., and Holmes, E.C., 1998, *Title Molecular evolution: a phylogenetic approach.* Blackwell Science, Oxford.

Panopoulou, G., Hennig, S., Groth, D., Krause, A., Poustka, A.J., Herwig, R., Vingron, M., and Lehrach, H., 2003, New Evidence for Genome-Wide Duplications at the Origin of Vertebrates Using an Amphioxus Gene Set and Completed Animal Genomes. *Genome Res.* 13: 1056-1066.

Paquola, A.C., Nishyiama, M.Y., Jr., Reis, E.M., da Silva, A.M., and Verjovski-Almeida, S., 2003, ESTWeb: bioinformatics services for EST sequencing projects. *Bioinformatics* 19: 1587-1588.

Pearson, W.R., and Lipman, D.J., 1988, Improved tools for biological sequence comparison. *Proc. Natl. Acad. Sci. USA* 85: 2444-2448.

Pearson, W.R., 1994, Using the FASTA program to search protein and DNA sequence databases. *Methods Mol Biol* 24: 307-331.

Pearson, W.R., 1998, Empirical statistical estimates for sequence similarity searches. *J Mol Biol* 276: 71-84.

Percudani, R., Pavesi, A., and Ottonello, S., 1997, Transfer RNA gene redundancy and translational selection in Saccharomyces cerevisiae. *J Mol Biol* 268: 322-330.

Perna, N.T., and Kocher, T.D., 1995, Patterns of nucleotide composition at fourfold degenerate sites of animal mitochondrial genomes. *J Mol Evol* 41: 353-358.

Perriere, G., Lobry, J.R., and Thioulouse, J., 1996, Correspondence discriminant analysis: a multivariate method for comparing classes of protein and nucleic acid sequences. *Comput Appl Biosci* 12: 519-524.

Pevzner, P.A., Borodovsky, M., and Mironov, A.A., 1989, Linguistics of nucleotide sequences. I: The significance of deviations from mean statistical characteristics and prediction of the frequencies of occurrence of words. *J Biomol Struct Dyn* 6: 1013-1026.

Pevzner, P.A., 2000, *Computational molecular biology: an algorithmic approach.* The MIT Press, Cambridge, Massachusetts.

Pevzner, P.A., Tang, H., and Waterman, M.S., 2001, An Eulerian path approach to DNA fragment assembly. *Proc. Natl. Acad. Sci. U S A* 98: 9748-9753.

Philippe, H., and Douady, C.J., 2003, Horizontal gene transfer and phylogenetics. *Curr Opin Microbiol* 6: 498-505.

Pickett, K.M., and Randle, C.P., 2005, Strange bayes indeed: uniform topological priors imply non-uniform clade priors. *Mol Phylogenet Evol* 34: 203-211.

Pielou, E.C., 1984, *The interpretation of ecological data: a primer on classification and ordination.* Wiley, New York.

Ponting, C.P., Schultz, J., Milpetz, F., and Bork, P., 1999, SMART: identification and annotation of domains from signalling and extracellular protein sequences. *Nucleic Acids Res* 27: 229-232.

Press, W.H., Teukolsky, S.A., Tetterling, W.T., and Flannery, B.P., 1992, *Numerical recipes in C: the art of scientifi computing.* Cambridge University Press, Cambridge.

Provitera, P., El-Maghrabi, R., and Scarlata, S., 2006, The effect of HIV-1 Gag myristoylation on membrane binding. *Biophys Chem* 119: 23-32.

Qin, Z.S., McCue, L.A., Thompson, W., Mayerhofer, L., Lawrence, C.E., and Liu, J.S., 2003, Identification of co-regulated genes through Bayesian clustering of predicted regulatory binding sites. *Nat Biotechnol* 21: 435-439.

Qu, K., McCue, L.A., and Lawrence, C.E., 1998, Bayesian protein family classifier. *Proc Int Conf Intell Syst Mol Biol* 6: 131-139.

Rabiner, L.R., 1989, A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE* 77: 257–286.

Raiffa, H., and Schlaifer, R., 1961, *Applied Statistical Decision Theory.* Division of Research, Graduate School of Business Administration, Harvard University.

Raoult, D., Audic, S., Robert, C., Abergel, C., Renesto, P., Ogata, H., La Scola, B., Suzan, M., and Claverie, J.M., 2004, The 1.2-megabase genome sequence of Mimivirus. *Science* 306: 1344-1350.

Rektorschek, M., Buhmann, A., Weeks, D., Schwan, D., Bensch, K.W., Eskandari, S., Scott, D., Sachs, G., and Melchers, K., 2000, Acid resistance of *Helicobacter pylori* depends on the UreI membrane protein and an inner membrane proton barrier. *Mol. Microbiol.* 36: 141-152.

Reyes, A., Gissi, C., Pesole, G., and Saccone, C., 1998, Asymmetrical directional mutation pressure in the mitochondrial genome of mammals. *Mol Biol Evol* 15: 957-966.

Rice, P., Longden, I., and Bleasby, A., 2000, EMBOSS: the European Molecular Biology Open Software Suite. *Trends Genet* 16: 276-277.

Rideout, W.M.I., Coetzee, G.A., Olumi, A.F., and Jones, P.A., 1990, 5-Methylcytosine as an endogenous mutagen in the human LDL receptor and p53 genes. *Science* 249: 1288-1290.

Robert-Seilaniantz, A., Shan, L., Zhou, J.M., and Tang, X., 2006, The pseudomonas syringae pv. tomato DC3000 Type III effector HopF2 has a putative myristoylation site required for its avirulence and virulence functions. *Mol Plant Microbe Interact* 19: 130-138.

Rocha, E.P., and Danchin, A., 2002, Base composition bias might result from competition for metabolic resources. *Trends Genet.* 18: 291-294.

Romero, A., and Garcia, P., 1991, Initiation of translation at AUC, AUA and AUU codons in Escherichia coli. *FEMS Microbiol Lett* 68: 325-330.

Rosenberg, M.S., and Kumar, S., 2003, Heterogeneity of nucleotide frequencies among evolutionary lineages and phylogenetic inference. *Mol Biol Evol* 20: 610-621.

Rosenblatt, F., 1958, The perceptron: a probabilistic model for information storage and organization in the brain. *Psychol Rev* 65: 386-408.

Rossi, F., and Conan-Guez, B., 2005, Functional multi-layer perceptron: a non-linear tool for functional data analysis. *Neural Netw* 18: 45-60.

Rouchka, E.C., 1997, A Brief Overview of Gibbs Sampling.: IBC Statistics Study Group, Washington University, Institute for Biomedical Computing.

Rouchka, E.C., Gish, W., and States, D.J., 2002, Comparison of whole genome assemblies of the human genome. *Nucleic Acids Res* 30: 5004-5014.

Rowe, D.C., McGettrick, A.F., Latz, E., et al., 2006, The myristoylation of TRIF-related adaptor molecule is essential for Toll-like receptor 4 signal transduction. *Proc. Natl. Acad. Sci. U S A* 103: 6299-6304.

Ruiz, L.M., Armengol, G., Habeych, E., and Orduz, S., 2006, A theoretical analysis of codon adaptation index of the Boophilus microplus bm86 gene directed to the optimization of a DNA vaccine. *J Theor Biol* 239: 445-449.

Ryan, M.J., Fox, J.H., Wilczynski, W., and Rand, A.S., 1990, Sexual selection for sensory exploitation in the frog *Physalaemus pustulosus*. *Nature* 343: 66-67.

Ryo, A., Kondoh, N., Wakatsuki, T., Hada, A., Yamamoto, N., and Yamamoto, M., 2000, A modified serial analysis of gene expression that generates longer sequence tags by nonpalindromic cohesive linker ligation. *Anal Biochem* 277: 160-162.

Sachs, G., Meyer-Rosberg, K., Scott, D.R., and Melchers, K., 1996, Acid, protons and *Helicobacter pylori*. *Yale J. Biol. Med.* 69: 301-316.

Sachs, G., Weeks, D.L., Melchers, K., and Scott, D.R., 2003, The gastric biology of *Helicobacter pylori*. *Annu. Rev. Physiol.* 65: 349-369.

Saha, S., Sparks, A.B., Rago, C., Akmaev, V., Wang, C.J., Vogelstein, B., Kinzler, K.W., and Velculescu, V.E., 2002, Using the transcriptome to annotate the genome. *Nat Biotechnol* 20: 508-512.

Saitou, N., and Nei, M., 1987, The neighbor-joining method: a new method for reconstructing phylogenetic trees. *Mol Biol Evol* 4: 406-425.

Sakaluk, S.K., 2000, Sensory exploitation as an evolutionary origin to nuptial food gifts in insects. *Proc Biol Sci* 267: 339-343.

Sakurai, N., and Utsumi, T., 2006, Posttranslational N-myristoylation is required for the anti-apoptotic activity of human tGelsolin, the C-terminal caspase cleavage product of human gelsolin. *J. Biol. Chem.* 281: 14288-14295.

Salzberg, S.L., Delcher, A.L., Kasif, S., and White, O., 1998, Microbial gene identification using interpolated Markov models. *Nucleic Acids Res* 26: 544-548.

Samso, M., Palumbo, M.J., Radermacher, M., Liu, J.S., and Lawrence, C.E., 2002, A Bayesian method for classification of images from electron micrographs. *J Struct Biol* 138: 157-170.

Sancar, A., and Sancar, G.B., 1988, DNA repair enzymes. *Annu. Rev. Biochem.* 57: 29-67.

Sankoff, D., Morel, C., and Cedergren, R.J., 1973, Evolution of 5S RNA and the non-randomness of base replacement. *Nat New Biol* 245: 232-234.

Sawa, T., and Ohno-Machado, L., 2003, A neural network-based similarity index for clustering DNA microarray data. *Comput Biol Med* 33: 1-15.

Schena, M., 1996, Genome analysis with gene expression microarrays. *Bioessays* 18: 427-431.

Schena, M., 2003, *Microarray analysis.* Wiley-Liss, New York.

Schmucker, D., Clemens, J.C., Shu, H., Worby, C.A., Xiao, J., Muda, M., Dixon, J.E., and Zipursky, S.L., 2000, Drosophila Dscam is an axon guidance receptor exhibiting extraordinary molecular diversity. *Cell* 101: 671-684.

Schultz, J., Copley, R.R., Doerks, T., Ponting, C.P., and Bork, P., 2000, SMART: a web-based tool for the study of genetically mobile domains. *Nucleic Acids Res* 28: 231-234.

Scott, D., Weeks, D., Melchers, K., and Sachs, G., 1998, The life and death of *Helicobacter pylori*. *Gut* 43: S56-60.

Scott, D.R., Marcus, E.A., Weeks, D.L., and Sachs, G., 2002, Mechanisms of acid resistance due to the urease system of *Helicobacter pylori*. *Gastroenterology* 123: 187-195.

Semple, C., and Steel, M., 2003, *Phylogenetics.* Oxford University Press, Oxford.

Seo, E.Y., Namkung, J.H., Lee, K.M., et al., 2005, Analysis of calcium-inducible genes in keratinocytes using suppression subtractive hybridization and cDNA microarray. *Genomics* 86: 528-538.

Shadel, G.S., and Clayton, D.A., 1997, Mitochondrial DNA maintenance in vertebrates. *Annu Rev Biochem* 66: 409-435.

Shannon, C.E., 1948, A Mathematical Theory of Communiction. *The Bell Systems Technical Journal* 27: 379-423.

Sharp, P.M., Tuohy, T.M., and Mosurski, K.R., 1986, Codon usage in yeast: cluster analysis clearly differentiates highly and lowly expressed genes. *Nucleic Acids Res* 14: 5125-5143.

Sharp, P.M., and Li, W.H., 1987, The codon Adaptation Index--a measure of directional synonymous codon usage bias, and its potential applications. *Nucleic Acids Res* 15: 1281-1295.

Shine, J., and Dalgarno, L., 1974, The 3'-terminal sequence of Escherichia coli 16S ribosomal RNA: complementarity to nonsense triplets and ribosome binding sites. *Proc. Natl. Acad. Sci. U S A* 71: 1342-1346.

Shine, J., and Dalgarno, L., 1975a, Terminal-sequence analysis of bacterial ribosomal RNA. Correlation between the 3'-terminal-polypyrimidine sequence of 16-S RNA and translational specificity of the ribosome. *Eur J Biochem* 57: 221-230.

Shine, J., and Dalgarno, L., 1975b, Determinant of cistron specificity in bacterial ribosomes. *Nature* 254: 34-38.

Siavoshi, F., Malekzadeh, R., Daneshmand, M., Smoot, D.T., and Ashktorab, H., 2004, Association between *Helicobacter pylori* Infection in gastric cancer, ulcers and gastritis in Iranian patients. *Helicobacter* 9: 470.

Sibler, A.P., Dirheimer, G., and Martin, R.P., 1986, Codon reading patterns in Saccharomyces cerevisiae mitochondria based on sequences of mitochondrial tRNAs. *FEBS Lett* 194: 131-138.

Siepel, A., and Haussler, D., 2004a, Combining phylogenetic and hidden Markov models in biosequence analysis. *J Comput Biol* 11: 413-428.

Siepel, A., and Haussler, D., 2004b, Phylogenetic estimation of context-dependent substitution rates by maximum likelihood. *Mol Biol Evol* 21: 468-488.

Siepel, A., and Haussler, D., 2005, Phylogenetic hidden Markov models. In *Statistical Methods in Molecular Evolution*. Nielsen, R. (ed). Springer, New York, pp. 325-351.

Sloane, A.J., Duff, J.L., Wilson, N.L., et al., 2002, High throughput peptide mass fingerprinting and protein macroarray analysis using chemical printing strategies. *Mol Cell Proteomics* 1: 490-499.

Smith, T.F., and Waterman, M.S., 1981a, Identification of common molecular subsequences. *J Mol Biol* 147: 195-197.

Smith, T.F., and Waterman, M.W., 1981b, Identification of common molecular subsequences. *J. Mol. Biol.* 147: 195-197.

Sneath, P.H.A., 1962, The construction of taxonomic groups. In *Microbial Classification*. Ainsworth, G.C. and Sneath, P.H.A. (eds). Cambridge University Press, Cambridge, pp. 289-332.

Sokal, R.R., and Michener, C.D., 1958, A statistical method for evaluating systematic relationships. *University of Kansas Sci. Bull.* 28: 1409-1438.

Solnick, J.V., Hansen, L.M., Salama, N.R., Boonjakuakul, J.K., and Syvanen, M., 2004, Modification of *Helicobacter pylori* outer membrane protein expression during experimental infection of rhesus macaques. *Proc. Natl. Acad. Sci. U S A* 101: 2106-2111.

Sparre, T., Larsen, M.R., Heding, P.E., Karlsen, A.E., Jensen, O.N., and Pociot, F., 2005, Unraveling the Pathogenesis of Type 1 Diabetes with Proteomics: Present And Future Directions. *Mol Cell Proteomics* 4: 441-457.

Stamm, S., Ben-Ari, S., Rafalska, I., Tang, Y., Zhang, Z., Toiber, D., Thanaraj, T.A., and Soreq, H., 2005, Function of alternative splicing. *Gene* 344: 1-20.

Stingl, K., Uhlemann Em, E.M., Deckers-Hebestreit, G., Schmid, R., Bakker, E.P., and Altendorf, K., 2001, Prolonged survival and cytoplasmic pH homeostasis of *Helicobacter pylori* at pH 1. *Infect. Immun.* 69: 1178-1180.

Stingl, K., Altendorf, K., and Bakker, E.P., 2002a, Acid survival of *Helicobacter pylori*: how does urease activity trigger cytoplasmic pH homeostasis? *Trends Microbiol.* 10: 70-74.

Stingl, K., Uhlemann, E.-M., Schmid, R., Altendorf, K., and Bakker, E.P., 2002b, Energetics of *Helicobacter pylori* and Its Implications for the Mechanism of Urease-Dependent Acid Tolerance at pH 1. *J. Bacteriol.* 184: 3053-3060.

Stormo, G.D., Schneider, T.D., Gold, L., and Ehrenfeucht, A., 1982a, Use of the 'Perceptron' algorithm to distinguish translational initiation sites in E. coli. *Nucleic Acids Res* 10: 2997-3011.

Stormo, G.D., Schneider, T.D., and Gold, L.M., 1982b, Characterization of translational initiation sites in E. coli. *Nucleic Acids Res* 10: 2971-2996.

Studier, J.A., and Keppler, K.J., 1988, A note on the neighbor-joining algorithm of Saitou and Nei. *Mol. Biol. Evol.* 5: 729-731.

Sueoka, N., 1961, Correlation bewteen base composition of deoxyribonucleic acid and amino acid composition of proteins. *Proc. Natl. Acad. Sci. USA.* 47: 1141-1149.

Suerbaum, S., Josenhans, C., Sterzenbach, T., et al., 2003, The complete genome sequence of the carcinogenic bacterium *Helicobacter hepaticus. Proc. Natl. Acad. Sci. U S A* 100: 7901-7906.

Sved, J., and Bird, A., 1990, The expected equilibrium of the CpG dinucleotide in vertebrate genomes under a mutation model. *Proc. Natl. Acad. Sci. USA.* 87: 4692-4696.

Swofford, D., 1993, Phylogenetic Analysis Using Parsimony. Champaign, IL: Illinois Natural History Survey.

Swofford, D.L., 2000, *Phylogeentic analysis using parsimony (* and other methods).* Sinauer, Sunderland, Mass.

Takahashi, N., Kaji, H., Yanagida, M., Hayano, T., and Isobe, T., 2003, Proteomics: advanced technology for the analysis of cellular function. *J Nutr* 133: 2090S-2096S.

Takezaki, N., and Nei, M., 1994, Inconsistency of the maximum parsimony method when the rate of nucleotide substitution is constant. *J. Mol. Evol.* 39: 210-218.

Tamayo, P., Slonim, D., Mesirov, J., Zhu, Q., Kitareewan, S., Dmitrovsky, E., Lander, E.S., and Golub, T.R., 1999, Interpreting patterns of gene expression with self-organizing maps: methods and application to hematopoietic differentiation. *Proc. Natl. Acad. Sci. USA.* 96: 2907-2912.

Tamura, K., and Nei, M., 1993, Estimation of the number of nucleotide substitutions in the control region of mitochondrial DNA in humans and chimpanzees. *Mol Biol Evol* 10: 512-526.

Tamura, K., and Kumar, S., 2002, Evolutionary distance estimation under heterogeneous substitution pattern among lineages. *Mol Biol Evol* 19: 1727-1736.

Tamura, K., Nei, M., and Kumar, S., 2004, Prospects for inferring very large phylogenies by using the neighbor-joining method. *Proc. Natl. Acad. Sci. U S A* 101: 11030-11035.

Tanaka, M., and Ozawa, T., 1994, Strand asymmetry in human mitochondrial DNA mutations. *Genomics* 22: 327-335.

Tang, N., Tornatore, P., and Weinberger, S.R., 2004, Current developments in SELDI affinity technology. *Mass Spectrom Rev* 23: 34-44.

Tao, H., Bausch, C., Richmond, C., Blattner, F.R., and Conway, T., 1999, Functional genomics: expression analysis of Escherichia coli growing on minimal and rich media. *J Bacteriol* 181: 6425-6440.

Tatusov, R.L., Koonin, E.V., and Lipman, D.J., 1997, A genomic perspective on protein families. *Science* 278: 631-637.

Tatusov, R.L., Fedorova, N.D., Jackson, J.D., et al., 2003, The COG database: an updated version includes eukaryotes. *BMC Bioinformatics* 4: 41.

Tavazoie, S., and Church, G.M., 1998, Quantitative whole-genome analysis of DNA-protein interactions by in vivo methylase protection in E. coli. *Nat. Biotechnol.* 16: 566-571.

Tavazoie, S., Hughes, J.D., Campbell, M.J., Cho, R.J., and Church, G.M., 1999, Systematic determination of genetic network architecture [see comments]. *Nat. Genet.* 22: 281-285.

Tech, M., and Merkl, R., 2003, YACOP: Enhanced gene prediction obtained by a combination of existing methods. *In Silico Biol* 3: 441-451.

Tenney, A.E., Brown, R.H., Vaske, C., Lodge, J.K., Doering, T.L., and Brent, M.R., 2004, Gene prediction and verification in a compact genome with numerous small introns. *Genome Res.* 14: 2330-2335.

Thayer, E.C., Olson, M.V., and Karp, R.M., 1999, Error checking and graphical representation of multiple-complete-digest (MCD) restriction-fragment maps. *Genome Res.* 9: 79-90.

Thijs, G., Lescot, M., Marchal, K., Rombauts, S., De Moor, B., Rouze, P., and Moreau, Y., 2001, A higher-order background model improves the detection of promoter regulatory elements by Gibbs sampling. *Bioinformatics* 17: 1113-1122.

Thijs, G., Marchal, K., Lescot, M., Rombauts, S., De Moor, B., Rouze, P., and Moreau, Y., 2002a, A Gibbs sampling method to detect overrepresented motifs in the upstream regions of coexpressed genes. *J Comput Biol* 9: 447-464.

Thijs, G., Moreau, Y., De Smet, F., Mathys, J., Lescot, M., Rombauts, S., Rouze, P., De Moor, B., and Marchal, K., 2002b, INCLUSive:

integrated clustering, upstream sequence retrieval and motif sampling. *Bioinformatics* 18: 331-332.

Thompson, J.D., Higgins, D.G., and Gibson, T.J., 1994, CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, positions-specific gap penalties and weight matrix choice. *Nucleic Acids Res.* 22: 4673-4680.

Thompson, W., Rouchka, E.C., and Lawrence, C.E., 2003, Gibbs Recursive Sampler: finding transcription factor binding sites. *Nucleic Acids Res* 31: 3580-3585.

Thompson, W., Palumbo, M.J., Wasserman, W.W., Liu, J.S., and Lawrence, C.E., 2004, Decoding human regulatory circuits. *Genome Res.* 14: 1967-1974.

Tian, Q., Stepaniants, S.B., Mao, M., et al., 2004, Integrated Genomic and Proteomic Analyses of Gene Expression in Mammalian Cells. *Mol Cell Proteomics* 3: 960-969.

Tomatsu, S., Orii, K.O., Bi, Y., et al., 2004, General implications for CpG hot spot mutations: methylation patterns of the human iduronate-2-sulfatase gene locus. *Hum Mutat* 23: 590-598.

Tomb, J.F., White, O., Kerlavage, A.R., et al., 1997, The complete genome sequence of the gastric pathogen *Helicobacter pylori*. *Nature* 388: 539-547.

Tong, K.L., and Wong, J.T., 2004, Anticodon and wobble evolution. *Gene* 333: 169-177.

Toronen, P., Kolehmainen, M., Wong, G., and Castren, E., 1999, Analysis of gene expression data using self-organizing maps. *FEBS Lett* 451: 142-146.

Trutschl, M., Dinkova, T.D., and Rhoads, R.E., 2005, Application of machine learning and visualization of heterogeneous datasets to uncover relationships between translation and developmental stage expression of C. elegans mRNAs. *Physiol Genomics* 21: 264-273.

Ukkonen, E., 1995, On-line Construction of Suffix Trees. *Algorithmica* 14: 249-260.

Valenzuela, M., Cerda, O., and Toledo, H., 2003, Overview on chemotaxis and acid resistance in *Helicobacter pylori*. *Biol. Res.* 36: 429-436.

Van Esch, H., and Devriendt, K., 2001, Transcription factor GATA3 and the human HDR syndrome. *Cell Mol Life Sci* 58: 1296-1300.

Vasilescu, J., and Figeys, D., 2006, Mapping protein-protein interactions by mass spectrometry. *Curr Opin Biotechnol* 17: 394-399.

Velculescu, V.E., Zhang, L., Vogelstein, B., and Kinzler, K.W., 1995, Serial analysis of gene expression. *Science* 270: 484-487.

Velculescu, V.E., Zhang, L., Zhou, W., Vogelstein, J., Basrai, M.A., Bassett, D.E., Jr., Hieter, P., Vogelstein, B., and Kinzler, K.W., 1997, Characterization of the yeast transcriptome. *Cell* 88: 243-251.

Velculescu, V.E., Madden, S.L., Zhang, L., et al., 1999, Analysis of human transcriptomes. *Nat. Genet.* 23: 387-388.

Venter, J.C., and Adams, M.D., and Myers, E.W., et al., 2001, The sequence of the human genome. *Science* 291: 1304-1351.

Vilas, G.L., Corvi, M.M., Plummer, G.J., Seime, A.M., Lambkin, G.R., and Berthiaume, L.G., 2006, Posttranslational myristoylation of caspase-activated p21-activated protein kinase 2 (PAK2) potentiates late apoptotic events. *Proc. Natl. Acad. Sci. U S A* 103: 6542-6547.

Viterbi, A.J., 1967, Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory* 13: 260–269.

Waddell, P.J., 1995, Statistical methods of phylogenetic analysis: including Hadamard conjugations, LogDet transforms, and maximum likelihood. Ph.D. thesis. New Zealand: Massey University.

Wang, H.C., and Hickey, D.A., 2002, Evidence for strong selective constraint acting on the nucleotide composition of 16S ribosomal RNA genes. *Nucleic Acids Res* 30: 2501-2507.

Wang, H.C., Singer, G.A., and Hickey, D.A., 2004, Mutational bias affects protein evolution in flowering plants. *Mol Biol Evol* 21: 90-96.

Wang, H.C., Xia, X., and Hickey, D.A., 2006, Thermal adaptation of ribosomal RNA genes: a comparative study. *J. Mol. Evol.* 63: 120-126.

Wang, J., Delabie, J., Aasheim, H., Smeland, E., and Myklebost, O., 2002, Clustering of the SOM easily reveals distinct gene expression patterns: results of a reanalysis of lymphoma study. *BMC Bioinformatics* 3: 36.

Wang, R.F., Campbell, W., Cao, W.W., Summage, C., Steele, R.S., and Cerniglia, C.E., 1996, Detection of *Pasteurella pneumotropica* in laboratory mice and rats by polymerase chain reaction. *Lab. Anim. Sci.* 46: 81-85.

Washburn, M.P., Wolters, D., and Yates, J.R., 3rd, 2001, Large-scale analysis of the yeast proteome by multidimensional protein identification technology. *Nat Biotechnol* 19: 242-247.

Waterfield, M.D., Scrace, G.T., Whittle, N., et al., 1983, Platelet-derived growth factor is structurally related to the putative transforming protein p28sis of simian sarcoma virus. *Nature* 304: 35-39.

Waterman, M.S., and Vingron, M., 1994, Rapid and accurate estimates of statistical significance for sequence data base searches. *Proc. Natl. Acad. Sci. U S A* 91: 4625-4628.

Weber, J.L., and Myers, E.W., 1997, Human whole-genome shotgun sequencing. *Genome Res.* 7: 401-409.

Weeks, D.L., Eskandari, S., Scott, D.R., and Sachs, G., 2000, A H+-gated urea channel: the link between *Helicobacter pylori* urease and gastric colonization. *Science* 287: 482-485.

Weiner, P., 1973, Linear Pattern Matching Algorithms. In *14th IEEE Annual Symposium on Switching and Automata Theory*, pp. 1-11.

Weir, B.S., 1990, *Genetic data analysis.* Sinauer Associates, Inc., Sunderland.

Wen, X., Fuhrman, S., Michaels, G.S., Carr, D.B., Smith, S., Barker, J.L., and Somogyi, R., 1998, Large-scale temporal gene expression mapping of central nervous system development. *Proc. Natl. Acad. Sci. USA.* 95: 334-339.

Wen, Y., Marcus, E.A., Matrubutham, U., Gleeson, M.A., Scott, D.R., and Sachs, G., 2003, Acid-adaptive genes of Helicobacter pylori. *Infect. Immun.* 71: 5921-5939.

Williams, C.L., Preston, T., Hossack, M., Slater, C., and McColl, K.E., 1996, *Helicobacter pylori* utilises urea for amino acid synthesis. *FEMS Immunol Med Microbiol* 13: 87-94.

Wilson, D.S., and Nock, S., 2002, Functional protein microarrays. *Curr Opin Chem Biol* 6: 81-85.

Wright, F., 1990, The 'effective number of condons' used in a gene. *Gene* 87: 23-29.

Wright, G.L., Jr., 2002, SELDI proteinchip MS: a platform for biomarker discovery and cancer diagnosis. *Expert Rev Mol Diagn* 2: 549-563.

Xia, X., 1996, Maximizing transcription efficiency causes codon usage bias. *Genetics* 144: 1309-1320.

Xia, X., Hafner, M.S., and Sudman, P.D., 1996, On transition bias in mitochondrial genes of pocket gophers. *J. Mol. Evol.* 43: 32-40.

Xia, X., 1998a, The rate heterogeneity of nonsynonymous substitutions in mammalian mitochondrial genes. *Mol Biol Evol* 15: 336-344.

Xia, X., 1998b, How optimized is the translational machinery in Escherichia coli, Salmonella typhimurium and Saccharomyces cerevisiae? *Genetics* 149: 37-44.

Xia, X., and Li, W.H., 1998, What amino acid properties affect protein evolution? *J Mol Evol* 47: 557-564.

Xia, X., 2000, Phylogenetic Relationship among Horseshoe Crab Species: The Effect of Substitution Models on Phylogenetic Analyses. *Syst. Biol.* 49: 87-100.

Xia, X., 2001, *Data analysis in molecular biology and evolution.* Kluwer Academic Publishers, Boston.

Xia, X., and Xie, Z., 2001a, AMADA: Analysis of microarray data. *Bioinformatics* 17: 569-570.

Xia, X., and Xie, Z., 2001b, DAMBE: Software package for data analysis in molecular biology and evolution. *J. Hered.* 92: 371-373.

Xia, X., and Xie, Z., 2002, Protein structure, neighbor effect, and a new index of amino acid dissimilarities. *Mol Biol Evol* 19: 58-67.

Xia, X., 2003, DNA methylation and mycoplasma genomes. *J. Mol. Evol.* 57: S21-S28.

Xia, X., 2004, A peculiar codon usage pattern revealed after removing the effect of DNA methylation. In *Fourth International Conference on Bioinformatics of Genome Regulation and Structure*. Vol. 1 Novosibirsk, Russia: IC&G, Novosibirsk, pp. 216-220.

Xia, X., 2005a, Content sensors based on codon structure and dna methylation for gene finding in vertebrate genomes. In

Bioinformatics Of Genome Regulation And Structure II. Kolchanov, N. and Hofestadt, R. (eds). Springer, pp. 21-29.

Xia, X., 2005b, Predicting protein production from transcriptomic data. In *The 11th International Conference on Information Systems Analysis and Synthesis*. Vol. 1. Aguilar, J., Chu, H.W., St-Amand, J., Galkin, I. and Chatzimisios, P. (eds.) Orland, Florida, USA: International Institute of Informatics and Systematics, pp. 7-10.

Xia, X., 2005c, Mutation and selection on the anticodon of tRNA genes in vertebrate mitochondrial genomes. *Gene* 345: 13-20.

Xia, X., and Palidwor, G., 2005, Genomic Adaptation to Acidic Environment: Evidence from *Helicobacter pylori*. *Am Nat* 166: 776-784.

Xia, X., and Yuen, K.Y., 2005, Differential selection and mutation between dsDNA and ssDNA phages shape the evolution of their genomic AT percentage. *BMC Genet* 6: 20.

Xia, X., 2006, Topological bias in distance-based phylogenetic methods: problems with over- and underestimated genetic distances. *Evolutionary Bioinformatics* 2: 375–387.

Xia, X., and Kumar, S., 2006, Codon-based detection of positive selection can be biased by heterogeneous distribution of polar amino acids along protein sequences. In *COMPUTATIONAL SYSTEMS BIOINFORMATICS: Proceedings of the Conference CSB 2006*. Vol. 4. Markstein, P. and Xu, Y. (eds). Imperial College Press, pp. 335-340.

Xia, X., Wang, H.C., Xie, Z., Carullo, M., Huang, H., and Hickey, D.A., 2006, Cytosine usage modulates the correlation between CDS length and CG content in prokaryotic genomes. *Mol Biol Evol* 23: 1450-1454.

Xia, X., 2007, Molecular phylogenetics: mathematical framework and unsolved problems. In *Structural approaches to sequence evolution.* Bastolla, U., Porto, M., Roman, H.E. and Vendruscolo, M. (eds). Springer, pp. 171-191.

Xia, X.H., Xie, Z., and Kjer, K.M., 2003a, 18S ribosomal RNA and tetrapod phylogeny. *Syst. Biol.* 52: 283-295.

Xia, X.H., Xie, Z., Salemi, M., Chen, L., and Wang, Y., 2003b, An index of substitution saturation and its application. *Mol. Phylogenet. Evol.* 26: 1-7.

Xiao, L., Wang, K., Teng, Y., and Zhang, J., 2003, Component plane presentation integrated self-organizing map for microarray data analysis. *FEBS Lett* 538: 117-124.

Yamaoka, Y., Kita, M., Kodama, T., Imamura, S., Ohno, T., Sawai, N., Ishimaru, A., Imanishi, J., and Graham, D.Y., 2002, *Helicobacter pylori* infection in mice: Role of outer membrane proteins in colonization and inflammation. *Gastroenterology* 123: 1992-2004.

Yang, M.Y., Bowmaker, M., Reyes, A., Vergani, L., Angeli, P., Gringeri, E., Jacobs, H.T., and Holt, I.J., 2002, Biased incorporation of ribonucleotides on the mitochondrial L-strand accounts for apparent strand-asymmetric DNA replication. *Cell* 111: 495-505.

Yang, Z., 1995, A space-time process model for the evolution of DNA sequences. *Genetics* 139: 993-1005.

Yang, Z., and Nielsen, R., 2000, Estimating synonymous and nonsynonymous substitution rates under realistic evolutionary models. *Mol Biol Evol* 17: 32-43.

Yang, Z., 2002, Phylogenetic analysis by maximum likelihood (PAML). Version 3.12. London: University College.

Yates, J.R., 2004a, Mass spectral analysis in proteomics. *Annu Rev Biophys Biomol Struct* 33: 297-316.

Yates, J.R., 2004b, Mass spectrometry as an emerging tool for systems biology. *Biotechniques* 36: 917-919.

Yip, T.T., and Lomas, L., 2002, SELDI ProteinChip array in oncoproteomic research. *Technol Cancer Res Treat* 1: 273-280.

Yokobori, S., Ueda, T., Feldmaier-Fuchs, G., Paabo, S., Ueshima, R., Kondow, A., Nishikawa, K., and Watanabe, K., 1999, Complete DNA sequence of the mitochondrial genome of the ascidian Halocynthia roretzi (Chordata, Urochordata). *Genetics* 153: 1851-1862.

Yokobori, S., Suzuki, T., and Watanabe, K., 2001, Genetic code variations in mitochondria: tRNA as a major determinant of genetic code plasticity. *J. Mol. Evol.* 53: 314-326.

Yokobori, S., Watanabe, Y., and Oshima, T., 2003, Mitochondrial genome of Ciona savignyi (Urochordata, Ascidiacea, Enterogona): comparison of gene arrangement and tRNA genes with Halocynthia roretzi mitochondrial genome. *J Mol Evol* 57: 574-587.

Yokobori, S., Oshima, T., and Wada, H., 2005, Complete nucleotide sequence of the mitochondrial genome of Doliolum nationalis with implications for evolution of urochordates. *Mol Phylogenet Evol* 34: 273-283.

Yokoyama, S., and Nishimura, S., 1995, Modified nucleotides and codon recognition. In *tRNA: structure, biosynthesis and function.* Soll, D. and RajBhandary, U. (eds). ASM Press, Washington, pp. 207-223.

Zhang, L., Zhou, W., Velculescu, V.E., Kern, S.E., Hruban, R.H., Hamilton, S.R., Vogelstein, B., and Kinzler, K.W., 1997, Gene expression profiles in normal and cancer cells. *Science* 276: 1268-1272.

Zhu, J., Liu, J.S., and Lawrence, C.E., 1998, Bayesian adaptive sequence alignment algorithms. *Bioinformatics* 14: 25-39.

Zon, L.I., Gurish, M.F., Stevens, R.L., Mather, C., Reynolds, D.S., Austen, K.F., and Orkin, S.H., 1991, GATA-binding transcription factors in mast cells regulate the promoter of the mast cell carboxypeptidase A gene. *J. Biol. Chem.* 266: 22948-22953.

Zwickl, D., and Holder, M., 2004, Model parameterization, prior distributions, and the general time-reversible model in Bayesian phylogenetics. *Syst Biol* 53: 877-888.

POSTSCRIPT

The thought of a reader reaching the end of a book always sends a thrilling feeling to the author. For me, it is like a sweet childhood dream come true. I used to have many dreams during my early childhood, in many colors and shades. They all turned into black and white when I reached nine years of age, when my father, a devoted communist falsely accused of being counterrevolutionary, died of torture in China, being a victim of abused power.

I have heard that many Chinese children today no longer dream about becoming authors of science. Instead, they want to become American president, and their parents would typically beam with pride when the youngsters expressed such ambitions. This has been terrifying to me. The world cannot afford to have many American presidents – just one seems to be damaging enough. It would seem more desirable for our younger generations to have modest dreams of writing books that are somewhat readable from the beginning to the end.

Dreaming of writing a book is nice and noble. Dreaming of becoming American president could be brutal and bloody. May the young minds not be corrupted by the evil of power!

Extreme power disrupts harmony, and harmony is the essence of life. We biologists know only too well that harmony manifests at all levels of organization of living beings, and disrupting harmony destroys not only the beauty of life, but also life itself. Yet harmony among different cellular components interacting within a cell and harmony among different cells interacting within an organism can only be achieved by these cells and cellular components following certain rules. If some cells break the rule, if they get out of the checking system, disasters such as cancer emerge, often with the consequence that either the misbehaving cell has to die or the organism has to die. Overtime multicellular organisms have evolved very complicated checking systems to safeguard against misbehaving cells. Life is charming with these checking systems, but turns ugly when such checking systems are broken.

Human communities and societies are not much different. People, including national leaders such as presidents and prime ministers, will want to gather more power to break the checking system. According to major religions, every individual has a sinful nature and has to be checked. Perhaps no one had better understanding of the sinful nature of human beings than the founding fathers of the United States of America who, based on this fundamental understanding, established a great political checking system that dramatically alleviated the detrimental effect of misbehaving leaders. They recognized every human to have a good half and a sinful half. The good half can be employed to accomplish public services, but the evil half deserves constant monitoring and checking.

How far has the present American administration deviated from the path chosen by the founding fathers! Instead of drawing a line within ourselves to recognize the good half and the sinful half, they draw a line among nations to recognize good nations and rogue nations. Everything good is “American” and everything else is “un-American”. How similar this is to former communist regimes where everything good was communist and everything bad was anti-communist!

Nature has created us, probably equal, but not perfect. We commit errors, seek forgiveness, and try to improve ourselves. The books we write are not perfect either. We solicit criticisms from our colleagues and seek forgiveness for serious errors and egregious omissions. We revise the books to improve them. We cleanse our souls to become better citizens. With the recognition of imperfection in us, we behave better, our books read better, and the world moves closer to perfection and harmony.

INDEX