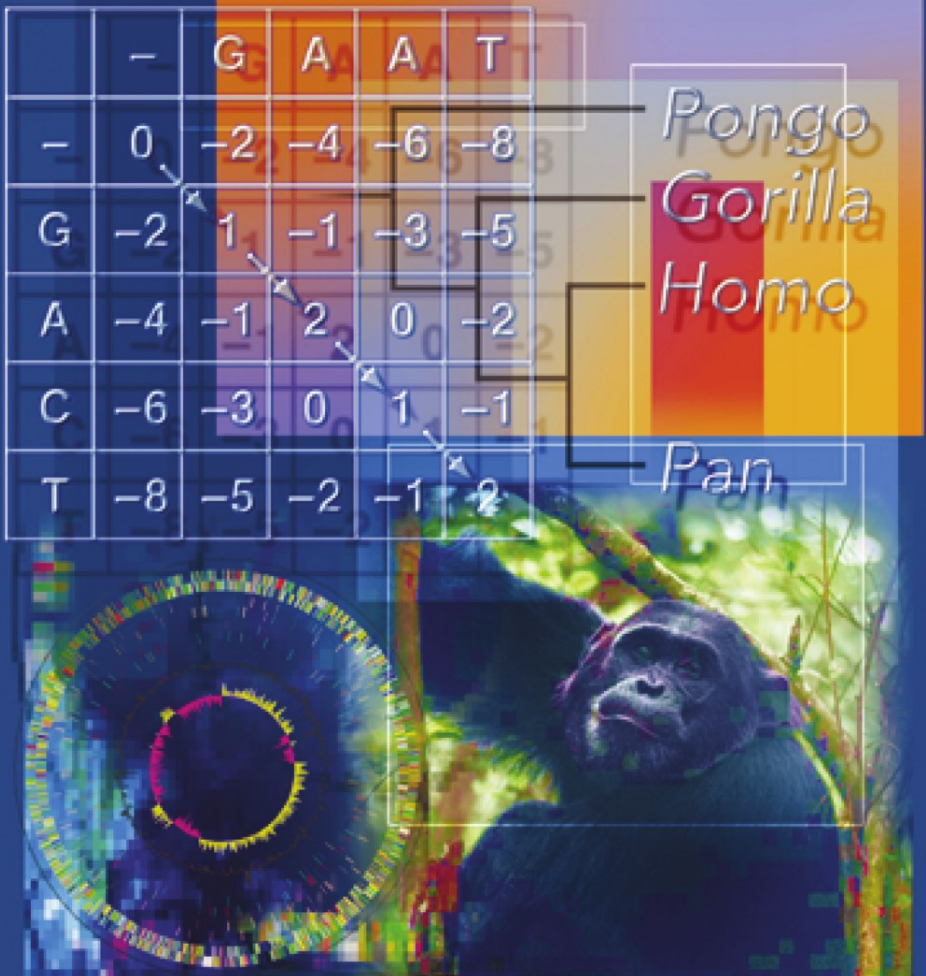


Richard C. Deonier
Simon Tavaré Michael S. Waterman

Computational Genome Analysis

An Introduction



Computational Genome Analysis

Richard C. Deonier, Simon Tavaré,
Michael S. Waterman

Computational Genome Analysis

An Introduction

With 102 illustrations, including 15 in full color

 Springer

Richard C. Deonier
Department of Biological Sciences
University of Southern California
Los Angeles, CA 90089
deonier@usc.edu

Michael S. Waterman
Department of Biological Sciences
University of Southern California
Los Angeles, CA 90089
msw@usc.edu

Simon Tavaré
Department of Biological Sciences
University of Southern California
Los Angeles, CA 90089
stavare@usc.edu
and
Department of Oncology
University of Cambridge
England

Cover collage: Lower left: Properties of the *Yersinia pestis* genome (see page 408 and also color insert). Reprinted, with permission from Parkhill J et al. (2003) *Nature* 413:523-527. Copyright 2003 Nature Publishing Group. Lower right: Photograph of chimpanzee by Professor Craig Stanford, Departments of Anthropology and Biological Sciences, University of Southern California.

Library of Congress Cataloging-in-Publication Data
Deonier, Richard C.

Computational genome analysis : an introduction / R.C. Deonier, S. Tavaré, M.S. Waterman.
p. cm.

Includes bibliographical references and index.

ISBN 0-387-98785-1 (alk paper)

I. Molecular Biology—Statistics—Computer Science. I. Tavaré, Simon.

II. Waterman, M.S. III. Title.

QH438.4M33W378 2005

572.8'6—dc22

2004059191

ISBN 0-387-98785-1

Printed on acid-free paper.

© 2005 Springer Science+Business Media, Inc.

All rights reserved. This work may not be translated or copied in whole or in part without the written permission of the publisher (Springer Science+Business Media, Inc., 233 Spring Street, New York, NY 10013, USA), except for brief excerpts in connection with reviews or scholarly analysis. Use in connection with any form of information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed is forbidden.

The use in this publication of trade names, trademarks, service marks, and similar terms, even if they are not identified as such, is not to be taken as an expression of opinion as to whether or not they are subject to proprietary rights.

Printed in the United States of America. (MVY)

9 8 7 6 5 4 3 2 1

SPIN 10715186

springeronline.com



R.C.D.:
To Sylvia, Rachel and Christian

S.T.:
To Jane, Ella and Richard

M.S.W.:
To the memory of Gian-Carlo Rota



Acknowledgments

We thank our students, particularly those who learned with us in BISC499 and BISC478. We also thank our colleagues in the Molecular and Computational Biology Section, Department of Biological Sciences, University of Southern California, for their comments and criticisms. We thank Professors Michelle Arbeitman, Lei Li, and Jeff Wall and Dr. Noah Rosenberg for providing illustrations, and Dr. Thomas D. Schneider, National Cancer Institute, for providing the sequence logo in Chapter 9.

We thank our editor, John Kimmel, for his patience and constructive criticisms during the writing of this book. We also thank the production editor Lesley Poliner and the production department at Springer (New York) for their valuable comments and assistance with formatting.

The cover photograph of the chimpanzee “Frodo” was kindly provided by Professor Craig Stanford, Departments of Anthropology and Biological Sciences, University of Southern California.

To the Student

This would normally be called a “preface,” but in our experience, students often skip sections with that title. We wanted to indicate what we assume about you, the reader, and how we think that you should use this book. First, we need to define what we mean by “student.” The need for this book became apparent to us when we were teaching an undergraduate computational biology course to seniors and first-year master’s or Ph.D. students at the University of Southern California. We have structured the book for use in an upper-level undergraduate teaching program. However, we recognize that “student” may be used in the broader sense to include anyone who wishes to understand the basics of computational biology. This might include investigators in medical schools or computer scientists who want to expand their knowledge of this exciting field.

Persons from a variety of disciplines develop an interest in computational biology. They may be biologists, applied mathematicians, computer scientists, and persons working in the biotechnology industry. Because of the variety of backgrounds from which students may be drawn, we cannot assume that any reader will have *all* of the necessary background. If we tried to supply all of the necessary background for all likely readers, this book would be prohibitively long (and expensive!). This means that you will probably need to supplement your reading with outside sources. For example, persons with backgrounds in applied mathematics may find Chapter 1 too telegraphic in style. They will need to use the supplementary biology texts indicated at the end of that chapter. Some current biology students may have limited computer skills and no programming skills. For them, we provide an appendix outlining the R statistics environment.

The latter point requires amplification. If you are studying computational biology, you should learn how to perform some simple computations. If you come from a computer science background, you will be proficient in UNIX and will be familiar with string-manipulation languages such as Perl and one or more versions of C. In contrast, some biology students may not know what commands can be issued at a UNIX prompt. How can we write a book that

employs computations that have the feel of “real” ones (computations beyond the hand calculator) without imposing a steep learning-curve barrier on those biology students? We have tried to solve this problem by using R.

R is a statistics environment that is available for free download and use with Windows, Macintosh, and Linux operating systems. It functions very much like the S-PLUS statistics package, which may be available to you already at your specific institution. R and S-PLUS provide an extensive suite of functions for statistics calculations. We are using R in lieu of C for computations because of its wide availability, because the input/output is simpler, and because the commands can be run interactively, which eases the writing of functions (programs). This means that you don’t need to first write a text file, compile the program, and then test it, going through this three-step process repeatedly until the program is debugged. However, if R is used for large-scale computations such as analysis of bacterial genomes (10^6 bp or more), be sure that you have available a large amount of memory or the computation time may be excessively long. For big problems, faster, more efficient programming languages such as versions of C can be incorporated into R.

You need to know how to actually implement the concepts in computational biology if you really want to understand them. Therefore, in many chapters we have included examples of computations using R. We encourage everyone to become familiar enough with R that the logic behind these computations is evident. We realize that this may be challenging for students lacking programming experience, but it really is necessary to learn how to visualize problems from a computational standpoint. To illustrate probabilistic models, we have performed a number of simulations. If you can’t visualize how to simulate data given a probabilistic model, do you really understand the model?

Finally, we emphasize that the material in this book does not represent the definitive word on any of the topics covered. It is not a treatise on statistics or algorithms in computational biology. It is not an overview of “bioinformatics,” although we treat some of the most important topics in that discipline. This book is not a compendium of useful Web sites with instructions on where you should point and click your mouse. However, in Appendix B we do provide some useful URLs to help you get started. This is not a book “about” genomics. To keep focused on the principles and strategies of the computations, we are not concerned with proving theorems and discussing the subtleties of the mathematics and statistics. If you are interested in more detail, you can refer to some of the literature cited at the end of the chapters. This book *is* a “roll up your sleeves and get dirty” introduction to the computational side of genomics and bioinformatics. We hope to provide you with a foundation for intelligent application of the available computational tools and for your intellectual growth as new experimental approaches lead to new computational tools.

We think that this field of endeavor is exciting and fun. Our hope is that we shall whet your appetite to learn more about what you are introduced to

here. In every case, there is much more already known and far more left to be discovered. We hope that you enjoy learning about it.

R.C.D., S.T., and M.S.W.
March 2005

Conventions and Resources

Terms in **bold type** are defined in the Glossary. Illustrations that are duplicated in the color insert are noted in the grey-scale versions appearing at the appropriate locations in the text. Computational examples throughout the text are enclosed in **grey boxes**. They are not required for the primary exposition of the material, but we encourage the reader to study the computational examples to see how the principles can be applied. Data sets and supplementary material are available for download at <http://www.cmb.usc.edu>.

Further Reading

A number of books for further reading are listed below. These are examples of books that we have found to be particularly useful. Other high-quality books have not been listed because of space limitations. Note that a variety of excellent monographs are available online at the National Center for Biotechnology Information (NCBI) Web site: <http://www.ncbi.nih.gov/entrez/query.fcgi?db=Books>.

Baxevanis AD, Ouellette BFF (2001) *Bioinformatics: A Practical Guide to the Analysis of Genes and Proteins* (2nd edition). New York: Wiley-Interscience.

Brown TA (ed) (2002) *Genomes* (2nd edition). New York: John Wiley & Sons.
Durbin R, Eddy SR, Krogh A, Mitchison G (1999) *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids*. Cambridge: Cambridge University Press.

Everitt BS, Dunn G (2001) *Applied Multivariate Data Analysis* (2nd edition). London: Arnold Publishers.

Ewens WJ, Grant GR (2001) *Statistical Methods in Bioinformatics*. New York: Springer-Verlag.

Strachan T, Read A (2003) *Human Molecular Genetics*, (3rd edition). New York: Garland Science/Taylor & Francis Group.

Contents

To the Student	ix
1 Biology in a Nutshell	1
1.1 Biological Overview	1
1.2 Cells	5
1.3 Inheritance	7
1.3.1 Mitosis and Meiosis	7
1.3.2 Recombination and Variation	9
1.3.3 Biological String Manipulation	12
1.3.4 Genes	13
1.3.5 Consequences of Variation: Evolution	17
1.4 Information Storage and Transmission	19
1.4.1 DNA	20
1.4.2 RNA	21
1.4.3 Proteins	21
1.4.4 Coding	24
1.5 Experimental Methods	25
1.5.1 Working with DNA and RNA	25
1.5.2 Working with Proteins	28
1.5.3 Types of Experiments	34
References	36
2 Words	37
2.1 The Biological Problem	37
2.2 Biological Words: $k = 1$ (Base Composition)	38
2.3 Introduction to Probability	40
2.3.1 Probability Distributions	40
2.3.2 Independence	42
2.3.3 Expected Values and Variances	43
2.3.4 The Binomial Distribution	44
2.4 Simulating from Probability Distributions	45

2.5	Biological Words: $k = 2$	48
2.6	Introduction to Markov Chains	50
	2.6.1 Conditional Probability	51
	2.6.2 The Markov Property	52
	2.6.3 A Markov Chain Simulation	54
2.7	Biological Words with $k = 3$: Codons	57
2.8	Larger Words	60
2.9	Summary and Applications	61
	References	62
	Exercises	62
3	Word Distributions and Occurrences	67
3.1	The Biological Problem	67
	3.1.1 Restriction Endonucleases	69
	3.1.2 The Problem in Computational Terms	70
3.2	Modeling the Number of Restriction Sites in DNA	71
	3.2.1 Specifying the Model for a DNA Sequence	71
	3.2.2 The Number of Restriction Sites	71
	3.2.3 Test with Data	73
	3.2.4 Poisson Approximation to the Binomial Distribution	74
	3.2.5 The Poisson Process	75
3.3	Continuous Random Variables	76
3.4	The Central Limit Theorem	79
	3.4.1 Confidence Interval for Binomial Proportion	81
	3.4.2 Maximum Likelihood Estimation	82
3.5	Restriction Fragment Length Distributions	84
	3.5.1 Application to Data	84
	3.5.2 Simulating Restriction Fragment Lengths	85
3.6	k -word Occurrences	89
	References	96
	Exercises	97
4	Physical Mapping of DNA	99
4.1	The Biological Problem	99
4.2	The Double-Digest Problem	101
	4.2.1 Stating the Problem in Computational Terms	101
	4.2.2 Generating the Data	101
	4.2.3 Computational Analysis of Double Digests	102
	4.2.4 What Did We Just Do?	104
4.3	Algorithms	105
4.4	Experimental Approaches to Restriction Mapping	106
4.5	Building Contigs from Cloned Genome Fragments	108
	4.5.1 How Many Clones Are Needed?	108
	4.5.2 Building Restriction Maps from Mapped Clones	110
	4.5.3 Progress in Contig Assembly	112

4.6	Minimal Tiling Clone Sets and Fingerprinting	115
	References	117
	Exercises	118
5	Genome Rearrangements	121
5.1	The Biological Problem	121
5.1.1	Modeling Conserved Synteny	123
5.1.2	Rearrangements of Circular Genomes	125
5.2	Permutations	126
5.2.1	Basic Concepts	126
5.2.2	Estimating Reversal Distances by Cycle Decomposition	129
5.2.3	Estimating Reversal Distances Between Two Permutations	131
5.3	Analyzing Genomes with Reversals of Oriented Conserved Segments	132
5.4	Applications to Complex Genomes	135
5.4.1	Synteny Blocks	135
5.4.2	Representing Genome Rearrangements	136
5.4.3	Results from Comparison of Human and Mouse Genomes	138
	References	140
	Exercises	140
6	Sequence Alignment	143
6.1	The Biological Problem	143
6.2	Basic Example	146
6.3	Global Alignment: Formal Development	152
6.4	Local Alignment: Rationale and Formulation	155
6.5	Number of Possible Global Alignments	157
6.6	Scoring Rules	160
6.7	Multiple Alignment	161
6.8	Implementation	163
	References	164
	Exercises	164
7	Rapid Alignment Methods: FASTA and BLAST	167
7.1	The Biological Problem	167
7.2	Search Strategies	169
7.2.1	Word Lists and Comparison by Content	170
7.2.2	Binary Searches	171
7.2.3	Rare Words and Sequence Similarity	171
7.3	Looking for Regions of Similarity Using FASTA	173
7.3.1	Dot Matrix Comparisons	173
7.3.2	FASTA: Rationale	173
7.4	BLAST	178

7.4.1	Anatomy of BLAST: Finding Local Matches	179
7.4.2	Anatomy of BLAST: Scores	180
7.5	Scoring Matrices for Protein Sequences	181
7.5.1	Rationale for Scoring: Statement of the Problem	182
7.5.2	Calculating Elements of the Substitution Matrices	183
7.5.3	How Do We Create the BLOSUM Matrices?	186
7.6	Tests of Alignment Methods	189
	References	190
	Exercises	192
8	DNA Sequence Assembly	195
8.1	The Biological Problem	195
8.2	Reading DNA	196
8.2.1	Biochemical Preliminaries	196
8.2.2	Dideoxy Sequencing	198
8.2.3	Analytical Tools: DNA Sequencers	202
8.3	The Three-Step Method: Overlap, Layout, and Multiple Alignment	203
8.4	High-Throughput Genome Sequencing	208
8.4.1	Computational Tools	209
8.4.2	Genome-Sequencing Strategies	212
8.4.3	Whole-Genome Shotgun Sequencing of Eukaryotic Genomes	214
	References	220
	Exercises	221
9	Signals in DNA	225
9.1	The Biological Problem	225
9.1.1	How Are Binding Sites on DNA Identified Experimentally?	226
9.1.2	How Do Proteins Recognize DNA?	227
9.1.3	Identifying Signals in Nucleic Acid Sequences	229
9.2	Representing Signals in DNA: Independent Positions	231
9.2.1	Probabilistic Framework	233
9.2.2	Practical Issues	236
9.3	Representing Signals in DNA: Markov Chains	242
9.4	Entropy and Information Content	248
9.5	Signals in Eukaryotic Genes	250
9.6	Using Scores for Classification	252
	References	259
	Exercises	260

10 Similarity, Distance, and Clustering	263
10.1 The Biological Problem	263
10.2 Characters	264
10.3 Similarity and Distance	268
10.3.1 Dissimilarities and Distances Measured on Continuous Scales	269
10.3.2 Scaling Continuous Character Values	271
10.4 Clustering	272
10.4.1 Agglomerative Hierarchical Clustering	272
10.4.2 Interpretations and Limitations of Hierarchical Clustering	276
10.5 K -means	278
10.6 Classification	286
References	286
Exercises	286
11 Measuring Expression of Genome Information	291
11.1 The Biological Problem	291
11.2 How Are Transcript Levels Measured?	292
11.3 Principles and Practice of Microarray Analysis	298
11.3.1 Basics of Nucleic Acids Used for Microarrays	298
11.3.2 Making and Using Spotted Microarrays	300
11.4 Analysis of Microarray Data	303
11.4.1 Normalization	303
11.4.2 Statistical Background	307
11.4.3 Experimental Design	310
11.5 Data Interpretation	313
11.5.1 Clustering of Microarray Expression Data	315
11.5.2 Principal Components Analysis	319
11.5.3 Confirmation of Results	320
11.6 Examples of Experimental Applications	321
11.6.1 Gene Expression in Human Fibroblasts	324
11.6.2 Gene Expression During <i>Drosophila</i> Development	324
11.6.3 Gene Expression in Diffuse Large B-cell Lymphomas	325
11.6.4 Analysis of the Yeast Transcriptome Using SAGE	327
11.7 Protein Expression	327
11.7.1 2DE/MALDI-MS	328
11.7.2 Protein Microarrays	329
11.8 The End of the Beginning	332
References	333
Exercises	334

12	Inferring the Past: Phylogenetic Trees	337
12.1	The Biological Problem	337
12.1.1	Example: Relationships Among HIV Strains	338
12.1.2	Example: Relationships Among Human Populations ...	338
12.1.3	Reading Trees	340
12.2	Tree Terminology	343
12.2.1	Conventions	343
12.2.2	Numbers of Trees	344
12.3	Parsimony and Distance Methods	346
12.3.1	Parsimony Methods	347
12.3.2	Distance Methods	350
12.4	Models for Mutations and Estimation of Distances	353
12.4.1	A Stochastic Model for Base Substitutions	354
12.4.2	Estimating Distances	355
12.5	Maximum Likelihood Methods	358
12.5.1	Representing a Tree	358
12.5.2	Computing Probabilities on a Tree	358
12.5.3	Maximum Likelihood Estimation	359
12.5.4	Statistics and Trees	360
12.6	Problems with Tree-Building	361
	References	361
	Exercises	363
13	Genetic Variation in Populations	367
13.1	The Biological Problem	367
13.2	Mendelian Concepts	368
13.3	Variation in Human Populations	369
13.3.1	Describing Variation Across Populations	370
13.3.2	Population Structure	374
13.4	Effects of Recombination	376
13.4.1	Relationship Between Recombination and Distance ...	378
13.4.2	Genetic Markers	379
13.5	Linkage Disequilibrium (LD)	381
13.5.1	Quantitative Description of LD	381
13.5.2	How Rapidly Does LD Decay?	383
13.5.3	Factors Affecting Linkage Disequilibrium	384
13.6	Linkage Disequilibrium in the Human Genome	386
13.7	Modeling Gene Frequencies in Populations	392
13.7.1	The Wright-Fisher Model	392
13.7.2	The Wright-Fisher Model as a Markov Chain	396
13.7.3	Including Mutation	397
13.8	Introduction to the Coalescent	398
13.8.1	Coalescence for Pairs of Genes	398
13.8.2	The Number of Differences Between Two DNA Sequences	400

13.8.3	Coalescence in larger samples	401
13.8.4	Estimating the Mutation Parameter θ	402
13.9	Concluding Comments	405
	References	405
	Exercises	407
14	Comparative Genomics	411
14.1	Compositional Measures	412
14.2	Transposable Elements	416
14.3	Sequence Organization within Chromosomes	418
14.3.1	Conservation of Synteny and Segmental Duplication	420
14.3.2	Identifying Conserved Segments and Segmental Duplications	422
14.3.3	Genome Evolution by Whole-Genome Duplication	425
14.4	Gene Content	432
14.4.1	Gene Prediction from Local Sequence Context	435
14.4.2	Exon and Intron Statistics	437
14.4.3	Comparative Methods for Identifying Genes	437
14.4.4	Gene Numbers	440
14.5	Predicted Proteome	441
14.5.1	Assigning Gene Function by Orthology	441
14.5.2	Assigning Gene Function by Patterns of Occurrence	443
14.5.3	Gene Content Within and Between Organisms	447
14.6	New Biological Perspectives from Genomics	452
	References	452
	Glossary	457
A	A Brief Introduction to R	479
A.1	Obtaining R and Documentation	479
A.2	First Steps	480
A.2.1	Starting, Stopping, and Getting Help	480
A.2.2	Objects	481
A.3	Types of Objects	482
A.4	Computations	487
A.5	Simple Statistical Applications	491
A.6	Functions	492
A.6.1	Writing Functions	492
A.6.2	Loops	493
A.6.3	Libraries	495
A.7	Graphics	496
A.7.1	Basic Plotting	496
A.7.2	Histograms	497
	References	498

B	Internet Bioinformatics Resources	499
B.1	General Entry Points	499
B.1.1	National Center for Biotechnology Information (NCBI)	499
B.1.2	European Bioinformatics Institute (EBI)	500
B.1.3	<i>Science</i> Magazine Functional Genomics Resources	501
B.2	Databases	501
B.3	Applications	504
B.4	Concluding Remarks	504
C	Miscellaneous Data	507
C.1	IUPAC-IUB Symbols	507
C.2	Genetic Code	508
C.3	<i>E. coli</i> Promoter Sequences	509
C.4	Yeast Gene Expression over Two Cell Cycles	511
C.5	Preprocessing of Microarray Data	512
	References	515

Biology in a Nutshell

The goal of computational genomics is the understanding and interpretation of information encoded and expressed from the entire genetic complement of biological organisms. The complete inventory of all DNA that determines the identity of an organism is called its **genome**. Biological systems are complicated, interacting multivariate networks. They can be considered at a number of different levels, ranging from populations of organisms to molecules. At the present time, computational biology emphasizes biological phenomena at levels of complexity ranging from molecular to cellular, though other levels in the hierarchy are also explored, especially in evolutionary contexts. The nature, anatomy, structure, physiology, biochemistry, and evolutionary histories of organisms define the types of problems to be solved. There are significant medical and evolutionary reasons to emphasize understanding human biology. Understanding the biology of other organisms, a worthy goal in its own right, also serves as a guide for interpreting the human genome and gene expression.

In this brief introduction we can only outline some key biological principles. For more details consult the monographs and Web sites listed at the end of the chapter.

1.1 Biological Overview

Zoos do not give a correct impression of what life on Earth is like because they over-represent mammals and other vertebrates. Organisms range from bacteria to multicellular plants and animals, and these organisms may employ a variety of strategies for extracting energy from their environment, ranging from reducing dissolved sulfate to produce H_2S and ultimately pyrite (fool's gold), to photosynthesis, to aerobic respiration. Some organisms can exist at temperatures near the boiling point (at atmospheric pressure) or below the freezing point of water. Others may be found in rocks 3 km below Earth's surface (lithotrophic bacteria) or flying over the Himalayas (snow geese). Nevertheless, analysis of ribosomal RNA sequences suggests that there are

three major related domains of organisms: **eubacteria** (organisms such as *Escherichia coli* or *Bacillus subtilis*), **Archaea** (bacteria notable for the extreme environments in which they can live), and **eukaryotes** (organisms with true nuclei, hierarchically structured chromosomes complexed with histones, and membrane-bound organelles—organisms such as humans or fungi). Relationships between these groups and between their representative members are indicated in Fig. 1.1. Two of the three major domains of life are **prokaryotic** (eubacteria and **archaeobacteria**).

Prokaryotes do not contain a true nucleus or membrane-bound organelles, and their DNA is not as highly structured as eukaryotic chromosomes. Given the wide range of environments bacteria can inhabit and their abundance from the ancient past up to the present, bacteria as a group are considered to be the most successful life form on this planet.

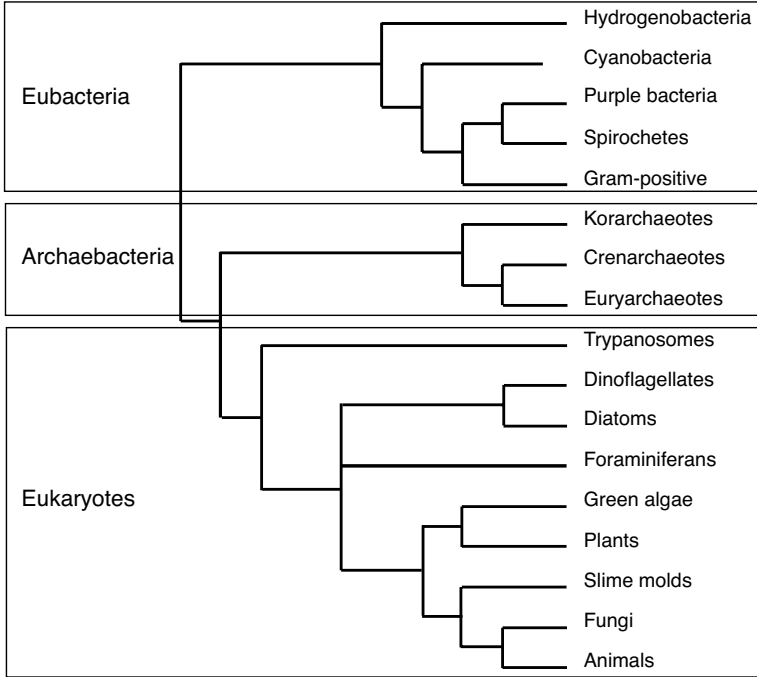
Among the eukaryotes, there is an abundance of unicellular forms, called protists. Most of these are marine organisms. Ultrastructural and molecular data indicate that different types of protists may differ from each other more than plants differ from animals. Nevertheless, the unicellular eukaryotes are conventionally lumped into a kingdom called “Protista.” Major multicellular groups are fungi, plants, and animals. There are about 300,000 described species of plants and about 1,000,000 described species of animals. This is a biased sample of the planet’s biodiversity. Among animals, mammals represent a rather small number of species. There are about 5000 species of mammals, but there are three times as many known species of flatworms. Three-quarters of all described animal species are insects. In terms of numbers of species, insects might be considered the most successful form of land animal.

There are similarities shared by all organisms on this planet:

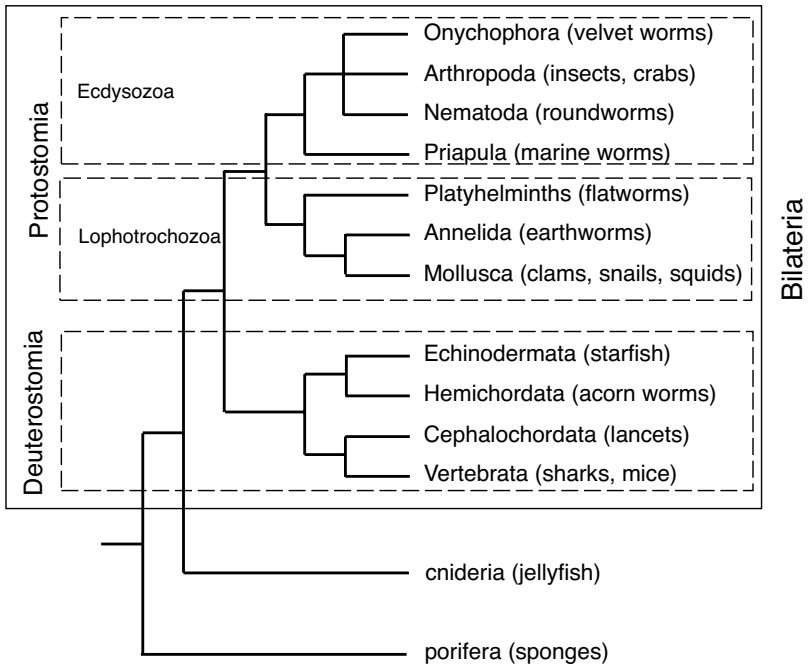
- The basic unit of life is the cell.
- Chemical energy is stored in ATP.
- Genetic information is encoded by DNA.
- Information is transcribed into RNA.

Fig. 1.1 (opposite page). Phylogenetic relationships among organisms (panel A) and among animals (panel B). Ancestor-descendant relationships are shown as a **tree** (see Chapter 12) with shared common ancestors corresponding to nodes to the left of descendant groups. The tree has been greatly simplified. Any given “twig” on the right can be further split to indicate descendant groups in more detail. There is usually a bifurcation at each node, but in those cases for which the branching order is unknown, there may be three (or more) descendant groups emanating from a particular node. Panel B indicates groupings of animals based upon body plans (bilateria), processes of embryological development (protostomes or deuterostomes), and physiological or anatomical features. Ecdysozoa shed their outer covering, lophotrochozoa share a type of feeding appendage or larval type, and chordata possess a notochord at some stage of development. Data from Pennisi (2003) and Knoll and Carroll (1999).

A.



B.



- There is a common triplet genetic code (with a few exceptions).
- Translation into proteins involves ribosomes.
- There are shared metabolic pathways (e.g., glycolysis), with steps catalyzed by proteins.
- Similar proteins are widely distributed among diverse groups of organisms.

These shared properties reflect the evolutionary relationships among organisms, which can be useful for understanding the significance of shared biological processes. For example, there are relationships between the pathways for bacterial photosynthesis with photosynthesis seen in cyanobacteria and plants. Some characters, such as the basic biochemical pathways, are so central to life that they are found in nearly all organisms. Processes such as replication, DNA repair, and glycolysis (extraction of energy by fermentation of glucose) are present and mechanistically similar in most organisms, and broader insights into these functions can be obtained by studying simpler organisms such as yeast and bacteria. It is unnecessary to start all over again from scratch in trying to understand functions encoded in genomes of new experimental organisms.

For efficient study, biologists have typically focused on model organisms that conveniently embody and illustrate the phenomena under investigation. Model organisms are chosen for convenience, economic importance, or medical relevance. Studies on such organisms are often applicable to other organisms that might be difficult to study experimentally. For example, the generation of antibody diversity in humans is equivalent to the process that can be genetically studied in mice. It was initially surprising to discover that developmental genes (hox genes) controlling segment specification in *Drosophila* (fruit flies) were mirrored by similar genes in mammals, including humans. Model organisms include bacteria such as *E. coli* and *B. subtilis* (now joined by many other bacteria whose genomes have been sequenced), fungi such as the yeasts *Saccharomyces cerevisiae* and *Schizosaccharomyces pombe*, simple animals such as the nematode *Caenorhabditis elegans*, insects such as *Drosophila melanogaster* (fruit fly), rapidly reproducing vertebrates such as *Danio rerio* (zebrafish) and mice (*Mus musculus*), and plants such as *Arabidopsis thaliana* (mustard weed). In addition to these are agriculturally important plants and animals (e.g., corn, or *Zea mays*) and of course humans (for medical reasons).

After this brief description of the complexity and scope of biological systems and organisms, in the rest of this chapter we will turn to those levels of complexity most pertinent to computational biology. First, we discuss cells, and we follow that with an introduction to informational macromolecules. We close by indicating some of the experimental methods that define the structure and scope of computational approaches.

1.2 Cells

Except for viruses, all life on this planet is based upon cells. Cells typically range in size from 2×10^{-6} m to 20×10^{-6} m in diameter (some cells, such as neurons, can be much larger). Cells sequester biochemical reactions from the environment, maintain biochemical components at high concentrations (which facilitates appropriately rapid reaction rates), and sequester genetic information. As mentioned above, structurally there are two different types of cells: prokaryotic and eukaryotic. **Prokaryotes** have cell membranes and cytoplasm, but the DNA is not separated from the cytoplasm by a nuclear membrane. Instead, the DNA is condensed into the *nucleoid*, which is less highly structured than eukaryotic chromosomes and appears as a disorganized “blob” in thin sections viewed by electron microscopy. Prokaryotes also lack membrane-bound organelles such as mitochondria and chloroplasts. Prokaryotic cells are typically small, and they may have generation, or doubling, times as short as 20–30 minutes. **Eukaryotes** (fungi, flies, mice, and men) have a true nucleus and membrane bound organelles. Most eukaryotes have observable **mitochondria**, where major steps in aerobic respiration occur. Plant cells may contain **chloroplasts**, where plant photosynthesis occurs. They also may have prominent vacuoles and cell walls composed of cellulose. The typical doubling time of eukaryotic cells from complex organisms is significantly longer than it is for prokaryotes: for a mammalian cell in tissue culture, this is about 24 hours (although some cells, such as neurons, may not divide at all).

Cells are organized into a number of components and compartments (Fig. 1.2). The plasma membrane—the “face” that the cell shows to the outside world—is decorated with transporter proteins capable of moving particular classes of molecules into and out of the cell. Because of their more complicated structure, eukaryotic cells have a more complex spatial partitioning of different biochemical reactions than do prokaryotes. For example, translation of particular mRNA molecules (ribonucleic acid copies of DNA coding for proteins) occurs on the endoplasmic reticulum, and processing of polypeptides may occur in the Golgi apparatus. The cellular **cytoskeleton** (composed of microtubules, microfilaments, and other macromolecular assemblages) aids in the trafficking of proteins and other cellular components from point to point in the cell. **Respiration** (the production of the energetic molecule ATP by oxidation of carbon compounds in the presence of oxygen) is localized on the membranes of mitochondria. All of these features imply that particular proteins may be localized for function in some compartments of the cell, but not others.

The simplest “food” requirements are seen with bacteria. For example, *E. coli* can grow in water containing ammonium chloride, ammonium nitrate, sodium sulfate, potassium phosphate, and magnesium sulfate (NH_4Cl , NH_4NO_3 , Na_2SO_4 , KH_2PO_4 , and MgSO_4 , respectively) at pH 7.2 with glucose as the sole source of carbon and energy. The water usually contains other

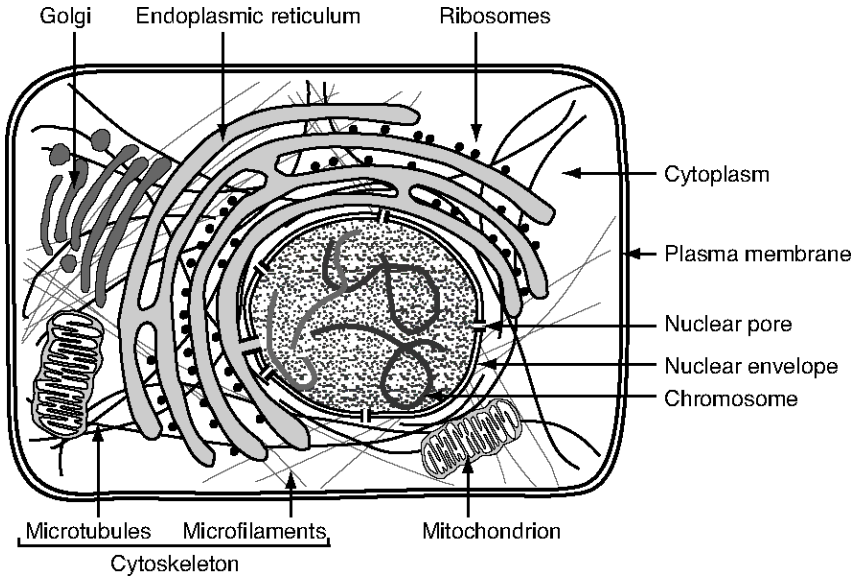


Fig. 1.2. Some major components of an animal cell (not necessarily drawn to scale). Some features (e.g., intermediate filaments, centrioles, peroxisomes) have not been depicted. In multicellular organisms, cells are frequently in contact and communication with other cells in tissues, but intercellular junctions and contacts with the extracellular matrix are not shown.

necessary metal ions in trace amounts. These substances flow into the cell through the inner and outer membranes. From a single type of sugar and inorganic precursors, a single bacterial cell can produce approximately 10^9 bacteria in approximately 20 hours; *E. coli* is also capable of importing other organic compounds into the cell from the outside, including other types of sugars and amino acids, when available.

To grow animal cells (e.g., human cells) in tissue culture, it is necessary to supply not only glucose and inorganic salts but also about a dozen amino acids and eight or more vitamins (plus other ingredients). Eukaryotic cells must import a large variety of components from the outside environment (matter flow). Because eukaryotic cells typically are 10 times larger in linear dimension than prokaryotic cells, their volumes are approximately 10^3 larger than volumes of prokaryotic cells, and diffusion may not suffice to move molecules into, out of, or through cells. Therefore, eukaryotic cells employ mechanisms of protein and vesicle transport to facilitate matter flow.

Another defining characteristic of eukaryotes is the machinery required for managing the genome during mitosis and meiosis (described below). Unlike prokaryotes, eukaryotes package their DNA in highly ordered chromosomes, which are condensed linear DNA molecules wrapped around octamers of proteins called histones. Since there are often many chromosomes, mechanisms

to ensure that each daughter cell receives a complete set are needed. This involves formation of the mitotic spindle, which includes microtubules that also contribute to the cell cytoskeleton. In addition, regulatory mechanisms are required to coordinate mitosis with DNA synthesis and the physiological state and size of the cell. These are fundamental processes that are shared by all eukaryotic cells.

This section has briefly presented a variety of information about the structure and biochemistry of cells. The DNA, RNA, and protein sequences with which computational biologists deal are important primarily because of the functions that they have within the cell. As we shall see, relating functions to macromolecules and sequences is one of the problems addressed in computational biology.

1.3 Inheritance

1.3.1 Mitosis and Meiosis

Each eukaryotic chromosome contains a single duplex DNA molecule bound with histone proteins to form a macromolecular complex. The sequences of bases contained on chromosomal DNA molecules are the result of a set of evolutionary processes that have occurred over time. These processes are intimately connected with how the chromosomes recombine and how they are copied during the DNA synthesis that precedes cell division.

Prokaryotes are typically **haploid** when they are not actively dividing, and they often (but not in every instance) have a single circular chromosomal DNA containing 10^6 – 10^7 bp (base pairs) of DNA. The DNA is typically inherited *vertically*, meaning that transmission is from parent to daughter cells. Under conditions of rapid growth or prior to cell division, there may be multiple copies of all or part of the prokaryotic chromosome, and except for low-frequency replication errors, their DNA sequences are usually identical. In such circumstances, recombination does not produce new assemblages of genes. Inheritance is *clonal* in the sense that descendants are more or less faithful copies of an ancestral DNA. This seemingly static mode of inheritance can be modified by transposable elements, by conjugation systems, and by acquisition of external DNA (transformation), but these interesting phenomena are beyond the scope of this introduction.

Sexual organisms such as mammals are usually **diploid**, which means that they contain N *pairs* of chromosomes (visible by light microscopy as stained chromatin). If the haploid chromosome number of an organism is N , the body (somatic) cells of that organism contain $2N$ chromosomes. There are two functional types of chromosomes: **autosomes**, which are not associated with sex determination, and sex chromosomes. Humans, for example, have 22 pairs of autosomes and two sex chromosomes: two X chromosomes for females, and one X + one Y for males. During the reproductive cycle of sexual organisms,

the **germline** tissues produce haploid sex cells, or **gametes**: ova from females and spermatozoa from males. Fusion of the gametes after mating produces a **zygote**, which will undergo development to form a new organism.

The sexual cycle involves an alternation between cells having $2N$ chromosomes or N chromosomes:

$$\begin{array}{rcl} \text{Parent 1: } 2N & \rightarrow & \text{Gamete 1: } N \\ & & + \\ & & \rightarrow \text{Zygote: } 2N \\ \text{Parent 2: } 2N & \rightarrow & \text{Gamete 2: } N \end{array}$$

The process of replication and reduction of chromosome numbers from $2N$ to N is called **meiosis**, which is confined to germline cells. Meiosis reduces the number of chromosomes by half because one chromosome doubling is followed by *two* cell divisions. Growth and development of the zygote is largely through a repeated process of chromosome doubling followed by one cell division—a process called **mitosis**. Cells destined to become germline cells are ordinarily subject to different sets of controls than typical body, or **somatic cells**. Mitosis of somatic cells is not genetically significant except for contributions that those cells may make to reproductive success (e.g., mitosis leading to colorful plumage in some male birds). Genetic mechanisms operate primarily during the formation and fusion of gametes.

Figure 1.3 follows two chromosomes during meiosis. Particularly important processes occur during prophase I, the beginning of the first meiotic division. As a result of the DNA synthesis that occurred during interphase, each chromosome has already been duplicated to generate a pair of *sister chromatids*. (Chromatids are precursors of chromosomes that have not yet been separated by meiosis.) Corresponding chromosomes from each parent (maternal and paternal copies of chromosome 7, for example) align with each other, and recombination occurs between corresponding maternal and paternal chromatids (i.e., between *nonsister* chromatids). **Recombination** is a process of breaking and joining chromosomes or DNA, and when this occurs at corresponding regions of a pair of similar molecules, the result is an exchange of these regions between the two molecules. This type of recombination is so-called *homologous recombination*—recombination between nearly identical sequences.

Overview of meiosis (See Fig. 1.3)

Step A: Chromatids from corresponding partner chromosomes from each parent recombine. Step B: Recombining chromosome partners (called bivalents) line up in the middle of the cell in preparation for the first meiotic cell division. Step C: Bivalents break apart, and one chromosome of each type moves to the opposite poles of the dividing cell. One or both chromatids may be recombinant. Step D: Completion of the first meiotic division produces two cells, each containing a haploid number of chromosomes, but each chromosome has two chromatids. Step E: Chromosomes line up at the center of the cell in preparation for the second meiotic division. Step F: During the second

meiotic division, bivalents in each duplicated chromosome are split, and one of each type is directed to one of the two daughter cells. The resulting cells are haploid with respect to chromosome number, and they contain only one genome equivalent.

Chromosomes are replicated only once, prior to prophase I. Thus there are four copies of each chromosome per cell at the beginning of a process that, through two cell divisions, will increase the number of cells by 2^2 . Metaphase I/anaphase I leads to separation of homologous *chromosomes*, while metaphase II /anaphase II leads to separation of *sister chromatids*. Recombination (prophase I) may involve multiple crossovers with both chromatids. Note that at anaphase I and anaphase II, chromosomes originating from one parent need not migrate to the same pole: assortment is independent and random. Only one of the meiosis II products becomes the egg in vertebrate females.

1.3.2 Recombination and Variation

Recombination between nonsister chromatids has extremely important genetic consequences. The frequencies and constraints of this process determine the **genetic map**, the **haplotypes**, and blocks of **conserved synteny**. (We will define these terms in the next paragraphs.) These are properties important in genetics, population genetics, and genome analyses. Each DNA or chromosome may contain alternative forms of given genes (an alternative form of a particular gene is called an **allele** of that gene). As a result of recombination during meiosis, the allele combinations in the gamete chromosomes are usually different from the combinations found in parental chromosomes. Thus, each gamete produced by parents drawn from a population represents a novel combination of alleles that were present in the population, and the resulting variation produced in successive generations is evolutionarily “tested” against the environment. Another source of variation is the production of new alleles by mutation (change in base sequence; see below). Moreover, it is possible for normal recombination processes to “derail,” leading to insertions, deletions, or duplications of longer stretches of chromosomal DNA sequence. These changes also are raw material for evolutionary change.

Chromosomes analyzed during genome projects display features that reflect recombination processes. One of the first tasks is to establish a correspondence between the DNA sequence and the genetic map. The **genetic map** records the order of genes and the approximate distances between them on their respective chromosomes. Genes are identified in classical genetics by particular mutations, sometimes called *genetic markers*. The order of genes is determined by *genetic crosses* (intentional mating of organisms having mutant genes), and the distances are measured in terms of recombination frequencies (often measured in **centimorgans**). A centimorgan corresponds to a recom-

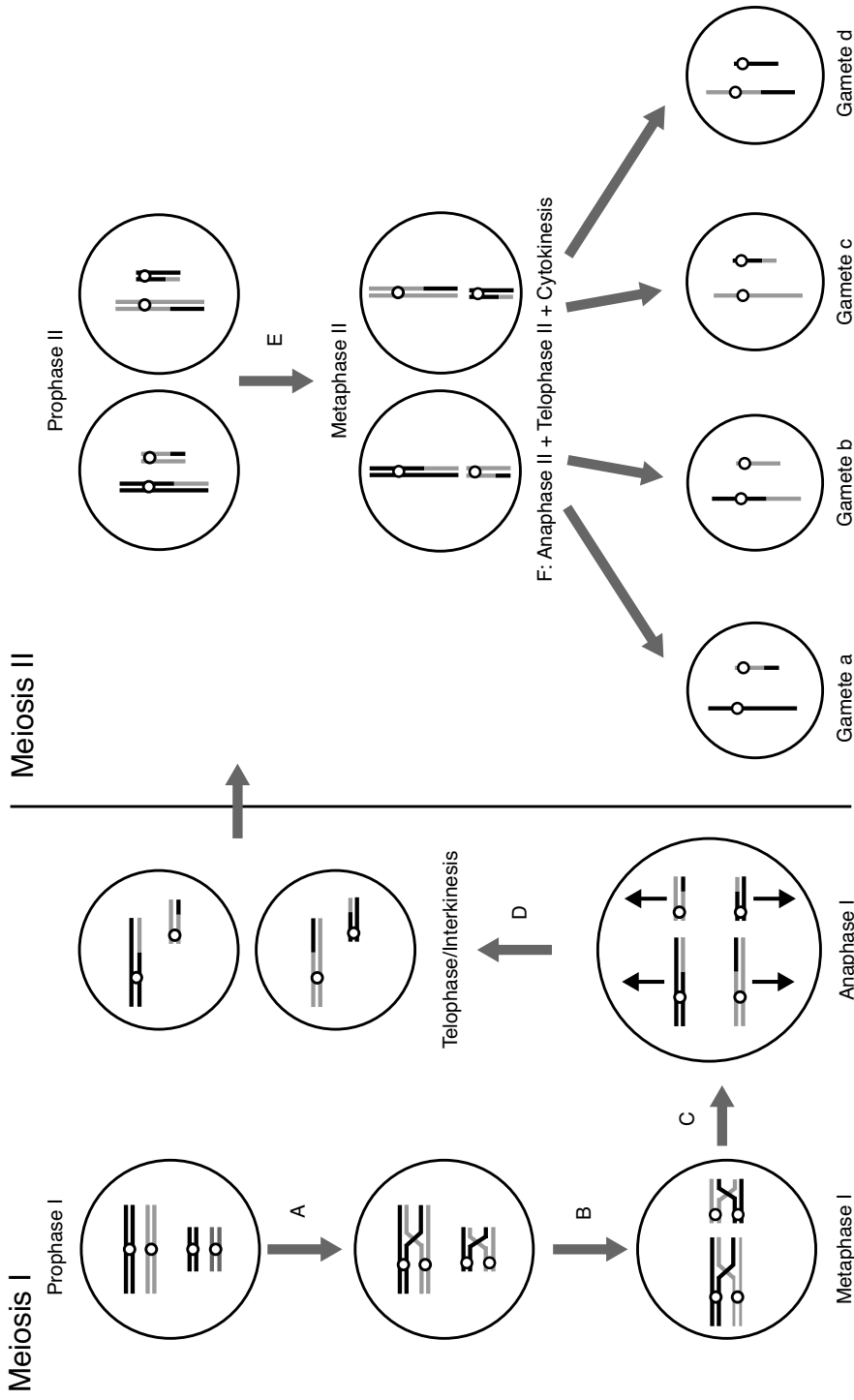


Fig. 1.3. Schematic summary of steps in meiosis (DNA replication and intermediate details not shown). In this diagram, the haploid chromosome number is 2 (one large and one small chromosome). Black chromosomes came from one parent, and grey chromosomes came from the other. For a description of processes A–F, see the accompanying box.

bination frequency of 1%, which means that two markers or genes that appear together on the same chromosome are separated from each other by recombination at a frequency of 0.01 during meiosis. Recombination is more likely to separate two distant markers than two close ones, and the recombination frequency between two markers is related to the physical distance separating them. Genes that tend to be inherited together are said to be genetically **linked**. If genetically linked alleles of several genes on a chromosome are so close together that they are rarely separated by recombination, this constellation of alleles may persist for a long period of time. Particular combinations of alleles carried on single chromosomes are called **haplotypes**, and frequencies of various haplotypes within a population characterize the structure of populations and can allow reconstruction of the evolutionary history of a population.

Over a longer timescale, recombination may shuffle the genetic maps of related species. For example, if species B and C are both descendants of ancestor A, the order of genes on the chromosomes of B and C might not be identical. Nevertheless, there may be groups of linked genes on a single chromosome in B and that also are linked on a particular chromosome of C. This circumstance is called **conserved synteny** (Fig. 1.4A; see Glossary for alternative definition). If the order of a set of genes is the same in both B and C, this set of genes is described as a **conserved segment**, and if high-density “landmarks” appear in the same order on a single chromosome in each of the two species, this set of landmarks defines a **syntenic segment**. (In some contexts, *conserved segments* and *syntenic segments* are also referred to as *conserved linkages* or *collinear gene clusters*). A set of adjacent syntenic segments is called a **syntenic block**, which may contain inversions and permutations of the syntenic segments of C compared with B (Fig. 1.4B). The numbers and sizes of such syntenic blocks are revealed when genome sequences of two organisms are compared, and these blocks are signatures of the evolutionary events separating B and C from A. It is possible to compare genomes of several related organisms and make inferences about their evolutionary relationships (i.e., comparative degrees of relatedness). One of the significant computational problems is the construction of phylogenetic trees based upon sequences or gene orders.

Even if there were no recombination, the DNA of the gametes would differ from the DNA of the parent cells because of errors that occur at low frequency during DNA replication. These errors occur at a frequency of 10^{-6} – 10^{-10} per base pair for each cell division (depending upon the cell, the genome, and the DNA polymerase involved). If the errors occur within a gene, the result may be a recognizable **mutation** (alteration of the base sequence in a normal gene or its control elements). Base changes at the DNA sequence level do not always lead to recognizable phenotypes, particularly if they affect the third position of a **codon** (three successive bases of DNA that encode a particular amino acid during translation). As a result of mutations occurring over time, a position in the DNA (whether in genes or in other regions of the genome)

A. Conserved synteny



B. Syntenic blocks and segments

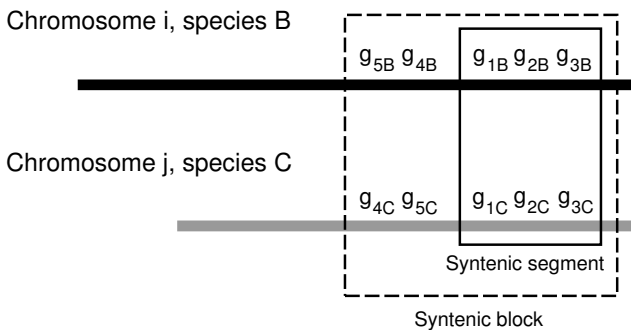


Fig. 1.4. Co-occurrence of genes or landmark sequences within single chromosomes or chromosome regions when chromosomes from each of two different organisms are compared. Panel A: Conserved synteny. In this case, g_{B1}, \dots, g_{B3} represent genes in species B that have homologs g_{C1}, \dots, g_{C3} in species C. Panel B: Syntenic segments and syntenic blocks. In this case, g_{B1}, \dots, g_{B5} and the similar sequences in species C refer to landmark sequences on the genome, which can be more numerous than genes to produce a higher marker density. Syntenic segments are conceptually similar to *conserved segments*, except that in the latter case there may be microrearrangements undetected because of the low marker density.

may contain different base pairs in different representatives of a population, and this variation can be measured at particular nucleotide positions in the genomes from many members of that population. This variation, when it occurs as an isolated base-pair substitution, is called a **single-nucleotide polymorphism**, or **SNP** (pronounced “snip”).

1.3.3 Biological String Manipulation

As indicated above, DNA is not immutable. During the copying or replication process, errors can occur (hopefully at low frequency, but at significantly high frequency in the case of reverse transcription of the HIV genome, for

example). In the human genome, the substitution rate at each nucleotide position averages $\sim 2.2 \times 10^{-9}$ per year (MGSC, 2002). The genome sequences of contemporary organisms contain a record of changes that have occurred over time. The types of changes that may have occurred include:

Deletion: Removal of one or more contiguous bases.

Insertion: Insertion of one or more contiguous bases between adjacent nucleotides in a DNA sequence. This is often associated with insertion of *transposable elements*.

Segmental duplication: Appearance of two or more copies of the same extended portion of the genome in different locations in the DNA sequence.

Inversion: Reversal of the order of genes or other DNA markers in a subsequence relative to flanking markers in a longer sequence. Within a longer sequence, inversion replaces one strand of the subsequence with its complement, *maintaining 5' to 3' polarity*.

Translocation: Placement of a chromosomal segment into a new sequence context elsewhere in the genome.

Recombination: In vivo joining of one DNA sequence to another. When similar sequences are involved, this is called *homologous recombination*.

Point mutation: Substitution of the base usually found at a position in the DNA by another as a result of an error in base insertion by DNA polymerase or misrepair after chemical modification of a base.

Results from some of these processes are diagrammed in Fig. 1.5. Point mutation is closely related to processes of DNA replication and repair for the generation or fixation of mutations. The other processes may also involve DNA copying, but they also involve other processes of DNA breaking and joining. Figure 1.6 makes an analogy between these processes and the menu for a computerized text editor, indicating the enzymes that may be involved in the various steps.

1.3.4 Genes

In the nineteenth century, Gregor Mendel observed that units of inheritance are discrete. A **gene** is now usually defined as a DNA segment whose information is expressed either as an RNA molecule or as a polypeptide chain (after translation of mRNA). Genes usually are found at defined positions on a chromosome, and such a position may be referred to as a **locus**. (A locus may correspond to a gene, but there are some loci that are not genes.)

Genes are identified biologically (i.e., in organisms studied in laboratories) when a mutation occurs that alters a phenotype or character. **Characters** are properties of an organism that can be observed or measured, and the phenotype corresponds to a particular state of a character. For example, a mutation in *Escherichia coli* may render the organism incapable of using lactose as a carbon source, or a mutation in *Drosophila melanogaster* may cause the eye

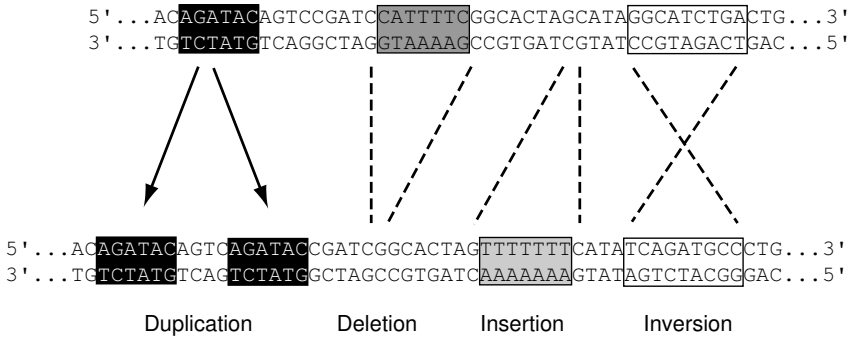


Fig. 1.5. Processes modifying multiple positions on a duplex DNA molecule. Although modifications of small numbers of basepairs are depicted, such modifications can involve much larger stretches of DNA (thousands to millions of bp or more). The processes named below the bottom molecule apply if the top molecule represents the starting condition. If the initial molecule were the one at the bottom, the DNA segment marked as “insertion” would be a “deletion” at the corresponding position in the top molecule. If it is unknown which molecule represents the initial state, such an insertion or deletion is called an “indel.”

Edit		<u>Process</u>	<u>Enzyme</u>
Cut		Cleave	Endonuclease
Copy	▶ As DNA As RNA	Replicate Transcribe	DNA polymerase RNA polymerase
Paste		Ligate	Ligase
Erase		Degrade	Exonuclease
Spelling		Repair or proofread	Repair enzymes or DNA polymerases

Fig. 1.6. The DNA text-editing “menu” (left) and associated enzymes.

color to change from red to white. In the latter case, the character is *eye color*, and the phenotype is *red eye color*. However, genes and phenotypes are not always in one-to-one correspondence. For example, in *E. coli* there are seven genes involved in the biosynthesis of the amino acid tryptophan from a precursor molecule. Mutations in any of these seven genes might lead to a Trp^- phenotype (which requires the addition of tryptophan for growth in the minimal medium). Similarly, a phenotype such as height or stature in *Homo sapiens* is controlled by a number of different genes, this time in a quantitative rather than all-or-none manner.

A given gene (corresponding to a particular locus) may have alternative forms called **alleles**, as described in Section 1.3.2. These alleles differ in DNA sequence, and these differences lead to differences in amino acid sequence. For example, individuals affected by sickle cell anemia have a beta-globin gene in which the glutamine normally present as the sixth amino acid residue is replaced by valine. This altered beta-globin gene is one allele of beta-globin, and the normal gene (wild-type gene) is another allele. There are other beta-globin alleles in the human population.

Genes are transcribed from DNA to form RNA molecules (including mRNA, a very important class of RNA). The DNA strand that is complementary to the mRNA is called the **template strand**, while the DNA strand whose sequence corresponds to the mRNA sequence is called the **coding strand**. DNA features (elements) found 5' relative to the coding sequence are considered to be "upstream," and elements that occur 3' relative to the coding sequence are referred to as "downstream." Diagrams of prokaryotic and eukaryotic genes are presented in Fig. 1.7.

Prokaryotic genes have a number of component elements. Going in the 5' to 3' direction relative to the direction of transcription, we encounter sites for binding proteins that control expression of the gene, a **promoter** where transcription initiates, the uninterrupted coding sequence (that eventually is translated into an amino acid sequence), and translational terminators. Sometimes the coding sequences for two or more polypeptide chains will be transcribed in succession from the same promoter, and such genes are said to be **polycistronic**. (*Cistrons* are identified by a particular type of genetic test, but they roughly correspond to coding sequences for polypeptide chains.) Polycistronic genes are relatively common in prokaryotes.

Eukaryotic genes are much more complicated than prokaryotic genes. They contain **exons**, which are segments of gene sequences that are represented in the processed mRNA. All segments of the coding sequence that will eventually be translated into protein appear in exons. **Introns** are noncoding DNA segments that separate the exons, and the RNA corresponding to introns is removed from the initial transcript by an excision process called **splicing**. Eukaryotic genes have more extensive control regions that include binding sites for transcription factors and enhancer sequences (that together regulate the level of transcription) and the "core" promoter where the transcription complex assembles. Eukaryotic transcription terminates nonspecifically down-

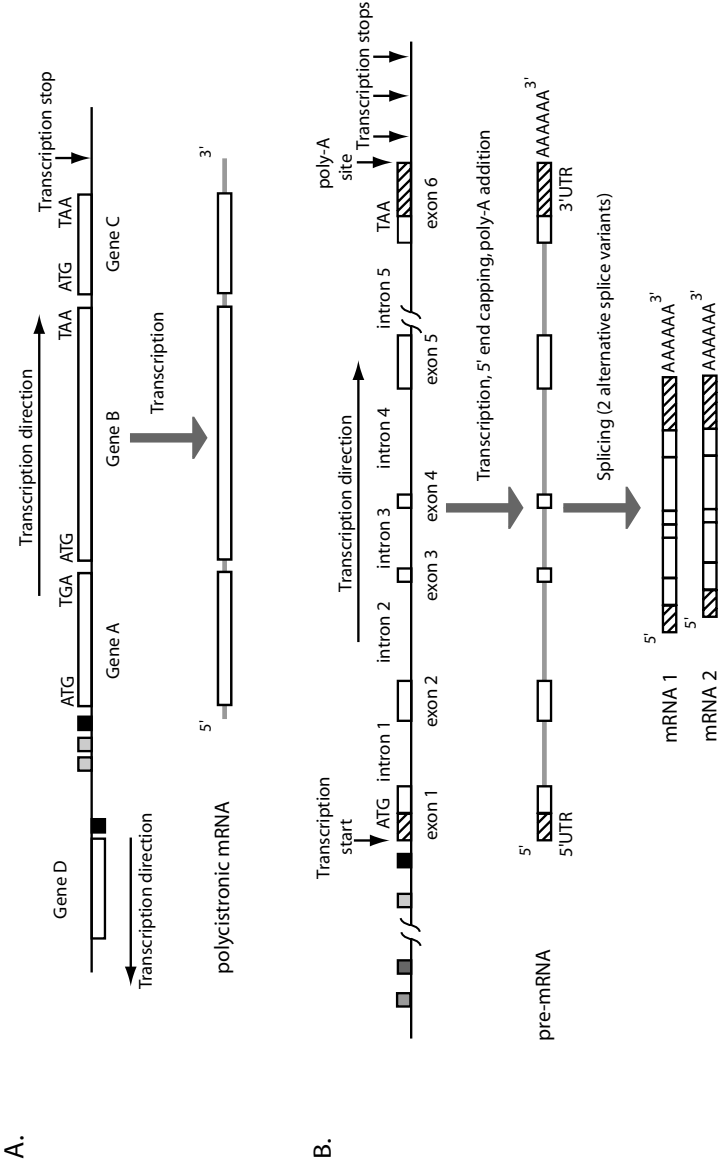


Fig. 1.7. Structures of prokaryotic and eukaryotic genes in duplex DNA (horizontal black line). White boxes above the solid black line correspond to coding sequences present on the “top” strand, while boxes below the line correspond to similar features on the “bottom” strand. Regulatory sequences (grey boxes) or core promoter sequences (black boxes) are indicated, but sizes and spacings are not to scale. DNA sequences for start and stop codons (ATG, UAG, UAA; UGA not shown) are indicated relative to the coding sequences. Panel A: Prokaryotic gene. Panel B: Eukaryotic gene. For a more complete description, see the accompanying box.

stream of the DNA “signal” that specifies the post-transcriptional addition of a string of A residues at the 3′ end of the message. (The addition of these A residues is called *polyadenylation*.)

Organization of prokaryotic and eukaryotic genes (See Fig. 1.7)

Prokaryotic genes

Genes in bacteria are normally close together and may be organized as operons (Genes A, B, and C) or may be individually transcribed (Gene D). Gene D is transcribed from the strand opposite to genes A, B, and C. Transcription (5′ to 3′ polarity with respect to the coding sequence) is initiated at promoter sequences, with initiation controlled by one or more operator sequences (grey boxes) to which repressor proteins bind. The mRNA product of the ABC operon (bottom of panel A) is ready for immediate translation.

Eukaryotic genes

The eukaryotic gene schematically depicted in panel B has transcription factor binding sites (grey boxes) upstream of the promoter. Promoter regions may be extensive, and introns may also be long, as indicated by the interruptions in the black line. The gene illustrated contains six exons and five introns. Exons 1 and 6 contain regions that will not be translated (5′ UTR in exon 1 and 3′ UTR in exon 6, hatched boxes). Transcription does not terminate at a unique position (vertical arrows). The immediate product of transcription is a pre-mRNA (grey line with open boxes) that is modified at the 5′ and 3′ ends. The poly-A tail is added to the end of the message after transcription has occurred. Splicing removes the introns to produce mature mRNA species that are ready for translation. Alternative splicing may or may not occur, but when it does, a single gene region may be responsible for two or more different (related) polypeptide chains (mRNA 1 and mRNA 2).

With the arrival of the “genome era,” genes can now be identified by analyzing DNA sequence. For prokaryotes, this may be relatively easy because the average prokaryotic gene is around 1000 bp long, and approximately 90% of a typical prokaryotic genome codes for gene products. For eukaryotes, this can be a significant computational problem because eukaryotic genes usually are much larger, are interrupted by several introns (in higher eukaryotes), and occur as a much smaller fraction of their genomes (around 1.2% in the case of *H. sapiens*). For example, the average human gene is about 27,000 bp in extent and contains 9–10 exons of average length 145 bp. The rest of the gene region corresponds to extensive control regions, untranslated regions, and the intronic regions, which may be thousands of base pairs in extent.

1.3.5 Consequences of Variation: Evolution

In the last two sections, we described types of change that can occur in DNA sequences, and we alluded to the biochemical mechanisms leading to these

changes. The result of such processes occurring in a large number of interbreeding individuals is genetic variation within the **population** (i.e., a localized collection of individuals in a species that reproductively transmits genes). This means that different versions (alleles) of the same gene may be found within the population. Some members of that species may be more reproductively successful than others, depending on their genetic constitution (**genotype**). Every organism's genotype is tested against environmental conditions through the phenotypes specified by that genotype. The conditions that a population experiences allow some genotypes to be more efficiently transmitted to the succeeding generations, leading to enrichment of some alleles at the expense of others. This process is called natural selection. The change in population gene or allele frequencies over time is called evolution.

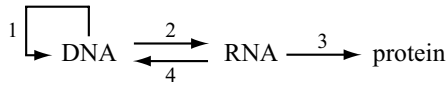
There are two related statistical and computational problems in dealing with populations: (1) characterization of genetic variation within and between populations in terms of allele frequencies or nucleotide sequence variation, and (2) analysis of the trajectory of population parameters over time. These two approaches are practically and conceptually interwoven. The first activity, which is the concern of population genetics, employs gene or locus frequency measurements taken from samples of populations alive today. Population genetics is usually (but not always) concerned with variation within species over relatively shorter periods of time. The second activity, known as molecular evolution, invokes evolutionary models to describe molecular (often sequence) data in a parsimonious manner. Molecular evolution usually (but not always) focuses on variation among species or higher-level taxa over comparatively longer periods of time. The key idea in evolutionary thought is that all species are related by descent from shared, common ancestral species that lived at some time in the past. These relationships are often represented in terms of phylogenetic trees (see Chapter 12). Thus, today's human and chimpanzee populations arose from the same ancestral primate population that existed about 6 million years ago, while humans and mice arose from an ancestral population that existed 80–85 million years ago (dates are estimated by a combination of molecular and fossil evidence). However, there were populations of organisms in the past that left no contemporary descendants (e.g., hadrosaurs, or “duck-billed” dinosaurs): they have become extinct. The biota of today are as much a result of extinction as of speciation. More than 99% of all species that have ever lived are now extinct. On average, species exist for about 2–4 million years before they succumb to extinction (some species last for much shorter, others for much longer times). Causes of extinction are varied and unpredictable. The organisms on Earth today resulted from a particular sequence of environmental conditions that occurred in a unique temporal sequence on this one planet. This pattern of evolution was not predictable and is not repeatable.

1.4 Information Storage and Transmission

An important segment of computational biology deals with the analysis of storage and readout of information necessary for the function and reproduction of cells. This information is used to code for structural proteins (e.g., cytoskeletal proteins such as actin and tubulin) and catalytic proteins (e.g., enzymes used for energy metabolism), for RNA molecules used in the translational apparatus (ribosomes, transfer RNA), and to control DNA metabolism and gene expression.

An organism's genome (defined at the beginning of the chapter) is, approximately, the entire corpus of genetic information needed to produce and operate its cells. As indicated above, eukaryotic cells may contain one, two, or three types of genomes: the nuclear genome, the mitochondrial genome, and the chloroplast genome (in plants). The vast majority of eukaryotes contain both nuclear and mitochondrial genomes, the latter being much smaller than the nuclear genome and confined to mitochondria. When one speaks of "the X genome" (e.g., "the human genome"), the nuclear genome is usually the one meant. Mitochondrial and chloroplast genomes in some ways resemble genomes of the prokaryotic symbionts from which they were derived.

The information flow in cells (already alluded to above) is summarized below.



The processes are identified as follows:

1. DNA replication, where a DNA sequence is copied to yield a molecule nearly identical to the starting molecule;
2. Transcription, where a portion of DNA sequence is converted to the corresponding RNA sequence;
3. Translation, where the polypeptide sequence corresponding to the mRNA sequence is synthesized;
4. Reverse transcription, where the RNA sequence is used as a template for the synthesis of DNA, as in retrovirus replication, pseudogene formation, and certain types of transposition.

Most biological information is encoded as a sequence of residues in linear, biological macromolecules. This is usually represented as a sequence of Roman letters drawn from a particular alphabet. Except for some types of viruses, DNA is used to store genomic information. RNA may be used as a temporary copy (mRNA) of information corresponding to genes or may play a role in the translational apparatus (tRNA, spliceosomal RNA, and rRNA). Proteins are polypeptides that may have catalytic, structural, or regulatory roles.

1.4.1 DNA

The genomes of free-living (nonviral) organisms are composed of DNA. The subunits (nucleotides) of these macromolecules are deoxyribonucleotides of four types: deoxyadenosine 5'-phosphate (A), deoxycytidine 5'-phosphate (C), deoxyguanosine 5'-phosphate (G), and thymidine 5'-phosphate (T). The 5' position on the sugar of each nucleotide is connected via a phosphate group to the 3' position on the sugar of the immediately preceding nucleotide. Each DNA strand has a 5' end, corresponding to the phosphate group attached to the 5' position on the sugar molecule of the first nucleotide, and a 3' end, corresponding to the -OH group at the 3' position on the sugar of the last nucleotide. For double-stranded DNA (Fig. 1.8), the two strands are antiparallel, which means that the two polynucleotide chains have opposite orientations or polarities. Base-pairing rules are usually observed: A base pairs with T and G base pairs with C (Fig. 1.9). Two strands whose sequences allow them to base pair are said to be complementary. A duplex DNA molecule can thus be represented by a string of letters drawn from {A, C, G, T}, with the left-to-right orientation of the string corresponding to the 5' to 3' polarity. The other strand is implied by the base-pairing rules. If the string corresponds to a single strand, then this should be explicitly stated. If a strand is written in the 3' to 5' direction, then this should be explicitly indicated. DNA molecules are encountered with numbers of bases or base pairs ranging from ~ 20 (oligonucleotide primers) to hundreds of millions (panel A of Table 1.1). For example, the DNA molecule in human chromosome 1 has a size of 285,000,000 base pairs. The number of bases or base pairs may be colloquially referred to as “length,” and units may be given in kilobases (kb = 1000 bases or base pairs) or megabases (Mb = 1,000,000 bases or base pairs).

The organization of DNA in cells can be considered at four different structural levels: constituent nucleotides, DNA, chromatin, and chromosomes. As an indicator of scale, the “length” of a nucleotide is approximately 1×10^{-9} m. The diameter of the DNA helix is 2×10^{-9} m, and the pitch of the helix is 3.4×10^{-9} m. In eukaryotes, the DNA is wrapped around histones to form nucleosomes (diameter 11×10^{-9} m). The chain of nucleosomes is further wrapped into higher-order structures that constitute the chromosomes, which are located in the nucleus. A typical nucleus might have a diameter of 0.5×10^{-5} m and would represent approximately 10% of the cell volume. Notice that a DNA molecule may be orders of magnitude longer than the diameter of the cell that contains it. For example, the length along the contour of the DNA in human chromosome 1 is approximately 9.5 cm (!), while the cell diameter is approximately 10^{-3} cm. The small diameter of the DNA helix and the hierarchical packing of the nucleosome structures allow packing of these long molecules into nuclei.

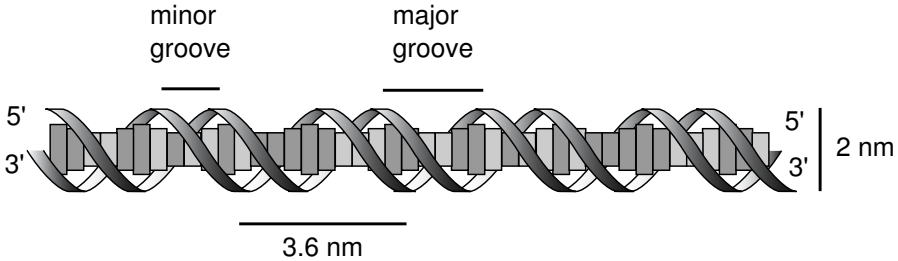


Fig. 1.8. Structure of duplex DNA. Ribbons represent the phosphodiester backbones of the two antiparallel strands, and the rectangular elements in the middle of the duplex represent the stacked base pairs. Connections of these base pairs to the phosphodiester backbone are not indicated. The gradients in size of the rectangles indicate that sometimes the base pairs are being viewed “edge-on” and other times “end-on” as they lie at different degrees of rotation about the helix axis. The major and minor grooves and relevant dimensions are indicated. Major and minor grooves are distinguished from each other by the spacing between the two phosphodiester backbones and the depth from the outside of the molecule to the edges of the base pairs.

1.4.2 RNA

RNA differs from DNA in two primary ways: the residues contain hydroxyl groups at the 2' position of the sugar (and thus are not “deoxy”), and uracil (U) replaces the thymine base T. Thus RNA molecules are composed of the monomers adenosine 5'-phosphate, cytidine 5'-phosphate, guanosine 5'-phosphate, and uridine 5'-phosphate. RNA is written as a string of Roman letters drawn from the alphabet {A, C, G, U}, with the left-to-right orientation corresponding to the 5' to 3' polarity. In most cases, RNA is encountered as a single strand, but often it will form intrastrand base pairs to form *secondary structures* that may be functionally important. RNA secondary structures are the result of intrastrand base pairing to form sets of “hairpins” and loops. The prediction of secondary structures for RNA molecules is a significant computational problem that takes into account free energies of base-pair formation and constraints on loop sizes. Duplex RNA molecules can be functional, either as genomes of some viruses or as interfering RNA (RNAi) that helps regulate gene expression in plants and animals. Sizes of RNA molecules typically range from approximately 100 to a few thousand nucleotides (not bp)—see panel B of Table 1.1.

1.4.3 Proteins

As indicated above, proteins are directly involved in the functioning of the cell. They are polypeptides, composed of strings of amino acid residues polymerized with the loss of one H₂O molecule per joined pair of amino acid residues. They

Table 1.1. Examples of DNA, RNA, and protein molecules. DNA molecules differ primarily by base composition and length and are structurally similar (but not identical) to each other. RNA molecules differ by length, base composition, and secondary and tertiary structure. Proteins are much more structurally diverse: the data represent only a sample of the diversity of structure types for proteins. Identical protein types from different organisms may differ in sequence; see the accession numbers for the source organism.

A: DNA

Name (GenBank acc. num.)	Number of bp	Base composition (%G + C)
Mouse mtDNA (NC_001569)	16,295	36.7
Bacteriophage λ (J02459)	48,502	49.9
<i>E. coli</i> K-12 chromosome (U00096)	4,639,221	50.8
Human chromosome 1	285,000,000	41.0

B: RNA

Name (GenBank acc. num.)	Number of nt	Base composition (%G + C)
tRNA _{Ala} (M26928)	73	60.3
18S rRNA(X04025)	1826	53.8
HIV-1 (AF443114)	9094	41.9

C: Protein

Name (PDB acc. num.)	Polypeptides (number/molecule)	Number of residues	Molecular weight
Ribonuclease A (1FS3)	A (1)	124	13,674
Total:	1	124	13,674
Hemoglobin (2HCO)	A (2)	141	15,110
	B (2)	146	15,851
Total:	4	574	61,922
Ubiquinol oxidase (1FFT)	A (1)	663	74,359
	B (1)	315	34,897
	C (1)	204	22,607
	D (1)	109	?
Total:	4	1291	148,000 (est)
Glutamine synthase (1FPY)	A (12)	468	51,669
Total:	12	5616	620,028

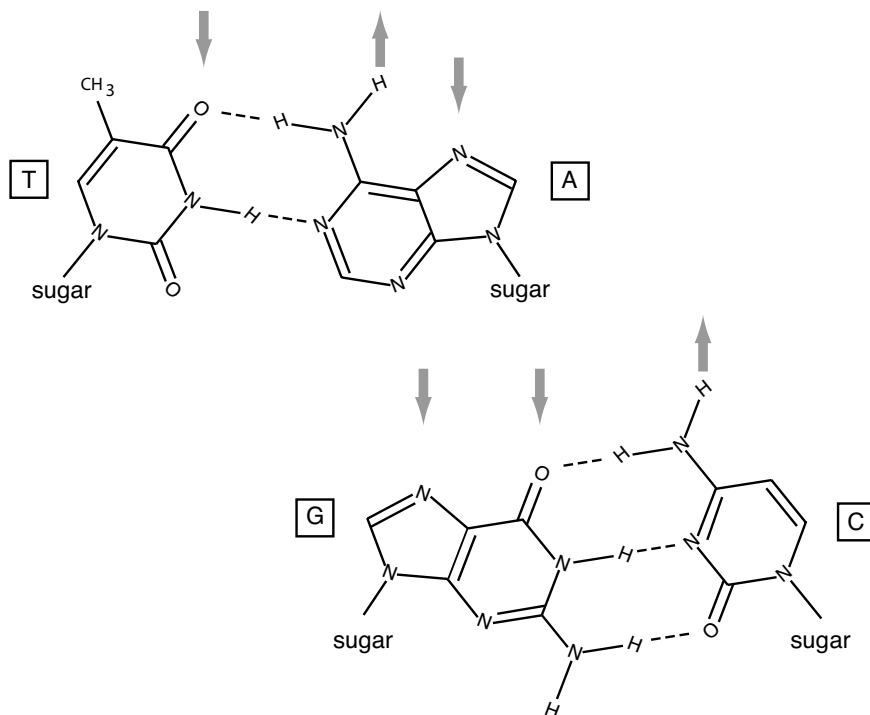
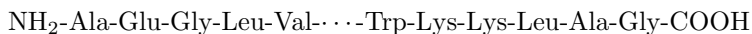


Fig. 1.9. Structure of Watson-Crick base pairs. Only relevant atoms are indicated. Junctions between straight segments in the rings correspond to locations of carbon atoms. The $-\text{CH}_3$ of T is called a methyl group. Broken lines connecting A and T or G and C correspond to hydrogen bonds that help stabilize the base pairs. Grey arrows pointing toward atoms on the rings indicate that these atoms are hydrogen bond acceptors, and grey arrows pointing away correspond to hydrogen bond donors. Only donors and acceptors on the major groove edges of the base pairs are shown. The bonds extending to the sugar residues in the phosphodiester backbone are on the minor groove side of the base pairs.

are usually represented as a string of letters drawn from an alphabet of twenty, written in the direction from the amino-terminal to the carboxy-terminal ends:



This also may be written as



using the three-letter abbreviation for the amino acid residues. Each polypeptide chain usually corresponds to a gene. Polypeptides in proteins usually have between 50 and 1000 amino acid residues, with 300 to 400 residues being the typical average length of polypeptides in many organisms. For small proteins (often in the range 100 to 200 amino acid residues), the active molecule may

be composed of a single polypeptide chain, but many proteins are composed of a precisely defined set of polypeptide chains. The simplicity of representation of polypeptides as a string of letters belies the profound structural complexity of protein molecules: the prediction of protein structure from the amino acid sequence is a difficult computational and theoretical problem.

Panel C of Table 1.1 lists some examples of proteins, illustrating the ranges in size and numbers of polypeptides. The sequence of amino acid residues constitutes the *primary structure*. There are a limited number of types of *secondary structures* (involving interactions between nearby amino acid residues on the same polypeptide chain), including alpha helix and beta pleated sheet. Secondary structure elements may combine to form a particular fold of a polypeptide segment. There are thought to be 1000–2000 different types of folds. Each individual polypeptide chain is folded into a particular three-dimensional structure (called *tertiary structure*). It is a general observation that complex proteins are often composed of multiple polypeptide subunits. Sometimes these are all identical, as is the case with glutamine synthase (12 identical polypeptide chains), but in other cases these subunits are all different (e.g., ubiquinol oxidase; see Table 1.1C). The aggregate structures formed from multiple polypeptide chains are called *quaternary structures*.

1.4.4 Coding

The DNA alphabet contains four letters but must specify polypeptide chains with an alphabet of 20 letters. This means that combinations of nucleotides are needed to code for each amino acid. Dinucleotides are combinations of two: AA, AC, AG, . . . , TC, TG, TT. There are 4^2 , or 16, possible dinucleotides—not enough to code for all 20 naturally occurring amino acids. Trinucleotides (triplets) are combinations of three nucleotides: AAA, AAC, AAG, . . . , TTC, TTG, TTT. There are 4^3 , or 64, possible trinucleotides. The genetic code is a triplet code, and the code triplets in mRNA are called **codons**. These may be written in their DNA form with T instead of U (when looking for genes in DNA) or in their RNA form with U instead of T (when we are concerned about the actual translation from mRNA). Triplets that specify “stop translation” are UAG, UGA, and UAA. Translational starts usually occur at an AUG codon, which also specifies the amino acid methionine. One representation of the genetic code is given in Appendix C.2.

Since there are three stop codons out of 64 triplets, there are 61 triplets coding for the 20 amino acids. This means that amino acids may be specified by more than one triplet. For example, leucine (Leu) is encoded by CUG, CUA, CUC, and CUU. As we will see later, these codons are not used with equal frequencies in various genes and organisms, and the statistics of codon usage is a characteristic that can sometimes be used to distinguish between organisms.

Successive amino acid residues in a polypeptide chain are specified by the sequence of triplets. For example, if the first amino acid is specified by a triplet beginning at nucleotide i in the mRNA, the second one will be specified by

the triplet beginning at nucleotide $i + 3$, and the third one will be specified by the triplet beginning at nucleotide $i + 6$, and so on. But what determines the location of the initial triplet at i ? Prokaryotes contain a hexanucleotide at the 5' end of the mRNA that sets the stage for translation beginning with the next AUG. Eukaryotes have a 5' cap structure, and translation begins at the next AUG of the mRNA.

When examining DNA for protein-coding regions, we initially look for **open reading frames** (ORFs). An **ORF** (pronounced “orf”) is a stretch of sequence that does not contain stop codons. In a random DNA sequence that is 50% G+C, on average one would expect a stop codon to occur after a stretch of 21 codons. The median exon size for humans is about twice as large as this (122 bp), and for prokaryotes the average gene size is approximately 1000 bp, so longer-than-average ORFs are indications of possible protein-coding regions. As a first approximation to gene recognition in prokaryotes, one looks for (1) an AUG start codon followed by (2) an open reading frame long enough to code for a reasonably sized polypeptide (> 50 amino acid residues) and having (3) the characteristic codon usage frequencies. The ORF ends at one of the three stop codons. As we will see later, gene recognition in eukaryotes requires much more analysis to replace point (2) above. Duplex DNA contains six possible reading frames, as illustrated in Fig. 1.10: three on the “top” strand and three on the “bottom” strand. When searching for genes in DNA, we must examine all six reading frames.

1.5 Experimental Methods

Data used in computational biology often can be traced back to a few microliters of an aqueous solution containing one or more types of biological macromolecules. The methods for studying biological materials determine the types of data available and the computational approaches required. The structures of DNA, RNA, and proteins were presented in Section 1.4. Here we discuss these types of molecules from an experimental perspective. As computational biologists, we should clearly understand what quantities are measured and how the measurements are made. It is appropriate to examine the raw data (e.g., output from a sequencing machine, autoradiogram, pattern of restriction fragments on a gel) to help understand the type and quality of the data.

1.5.1 Working with DNA and RNA

Most DNA and RNA molecules, regardless of their source, have similar properties and can be purified by using only minor variations from a standard set of protocols: alkaline minipreps (appropriate for DNA, not RNA), ultracentrifugation in CsCl gradients, or ethanol precipitations, for example. The experimental approaches depend on the amounts and molecular sizes of each type of macromolecule in a typical cell.

Start Codon: AUG (ATG in DNA) Stop Codons: UAA (TAA in DNA)
 UAG (TAG in DNA)
 UGA (TGA in DNA)

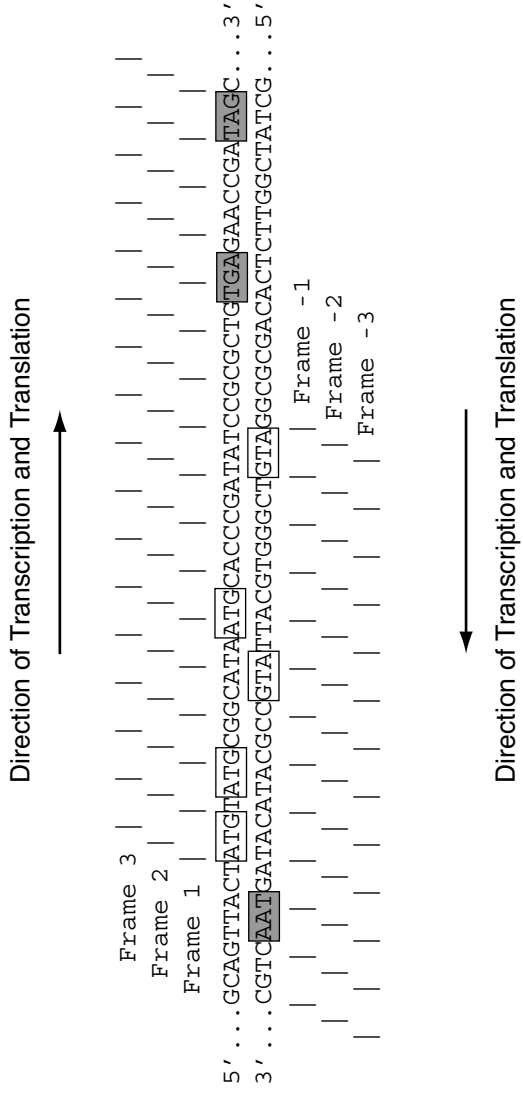


Fig. 1.10. Relationships among open reading frames, coding or “sense” strands, and template strands in duplex DNA. For frames 1, 2, and 3 above the DNA sequence, the “top” strand is the coding strand and the bottom strand is the template. For the frames written below the sequence, the situation is reversed. For actual open reading frames in prokaryotes, the number of codons is, on average, approximately 300. Processed eukaryotic mRNA molecules are defined largely by their exons, which average approximately 50 codons in humans.

We can calculate the abundance of a 1000 bp segment of single-copy DNA in diploid eukaryotic cells. Taking each base pair of the sodium salt of DNA to have a molecular mass of 662, and recognizing that there are two copies of each molecule per cell, we calculate that 10^3 bp of single-copy DNA in a genome corresponds to about 2×10^{-18} grams of DNA/cell. A 1 liter culture of mammalian cells grown in suspension (at about 10^6 cells/mL in a tissue culture flask, higher in a bioreactor) would contain 2×10^{-9} g of this 1000 bp region. In contrast, a DNA segment of similar length in mitochondrial DNA (mtDNA, present at 10^3 - 10^4 copies/mammalian somatic cell) will be at least a thousand times more abundant. The molecular sizes also matter. A 1000 bp DNA segment of eukaryotic DNA from a particular chromosome is part of a long, linear DNA molecule that cannot be easily purified without fragmentation, which destroys long-distance relationships to other regions of the molecule. In contrast, a similar length segment in mitochondrial DNA can be easily purified on an intact molecule because the mtDNA molecules are small, circular molecules that can be purified without fragmentation.

For routine molecular biology procedures (e.g., restriction mapping, in vitro mutagenesis), a laboratory technician requires about 10^{-7} to 10^{-8} g of DNA. Because only small quantities of any particular nuclear DNA sequence are isolated directly (see above), DNA is often amplified. The most common amplification methods are the polymerase chain reaction (**PCR**) and cloning. PCR employs in vitro enzymatic DNA synthesis in repeated cycles, such that if amount A of a DNA is originally present, after n cycles the amount present will be approximately $A2^n$. With extreme care to exclude contaminating DNA, it is technically possible to amplify as little as one molecule (!) by PCR. There is no DNA repair system to correct polymerase copying errors during PCR; consequently, after repeated cycles of amplification, some of the copies will have slightly altered sequences compared with the original, unamplified molecules.

Cloning (see Fig. 1.11) involves enzymatic joining of the desired DNA sequence to a cloning vector and its propagation in a convenient host organism (bacteria, yeast, or eukaryotic cells in culture). The **cloning vector** is a DNA molecule chosen to have appropriate replication properties (copy number, control of copy number), insert size, and host specificity (vectors must be matched to the host cell) for producing the desired amount and length of DNA product. If a genome is being analyzed, the genome may be represented as collections of clones, called **libraries**. In genome sequencing projects, clone collections based upon three different types of cloning vector are not uncommon. The number of clones to be collected and stored may range from 10^4 to 10^7 , depending upon the genome size of the organism being analyzed, the chosen vector, and the cloned fragment size. This of course raises practical issues of physically archiving the clones (e.g., sample storage in freezers) and recording pertinent descriptive information about each (e.g., cloning vector, date prepared, shelf location in a particular freezer).

Microarray studies of gene expression focus on types and amounts of different RNA species in cells or tissues (see Chapter 11). Mammalian genomes may have $\sim 25,000$ genes, which corresponds to 25,000–75,000 or more possible different mRNA species when alternative splice variants are taken into account. Different mRNA species can have very different levels of abundance, and they are often unstable (some may be degraded *in vivo* within seconds to minutes). mRNA molecules are extremely unstable *in vitro* as well. Unlike DNA, they are sensitive to alkaline hydrolysis. Stable ribonucleases present in cells (and on laboratory apparatus and fingerprints of technicians who don't wear protective gloves) represent an even bigger problem. Sequences represented on mRNA molecules can be enzymatically converted to DNA form by *in vitro* reverse transcription using retroviral reverse transcriptases. Since the DNA strands that are reverse-transcribed are complementary to the mRNA molecules, these products are referred to as **cDNA** (Fig. 1.11, right). Such molecules may not contain copies of the complete mRNA sequence, however. The molecules are usually converted to duplex cDNA for eventual manipulation (e.g., cloning). Collections of cloned cDNAs representing transcripts from a particular cell type are called cDNA libraries. Of course, cDNA molecules can be further amplified by PCR as required.

Sometimes short DNA sequences (~ 200 nucleotides) are obtained from large collections of cDNA clones to provide sequence labels for genes expressed under a particular set of conditions for a particular cell or tissue type. These short sequences are called expressed sequence tags (**ESTs**). The sequences of ESTs can be used to design PCR primers that can assist in mapping these sequences to genomic locations.

1.5.2 Working with Proteins

Different proteins can have substantially different abundances. For example, proteins such as histones or cytoskeletal proteins are abundant in cells and are readily purified. Other proteins have very low abundances (< 100 molecules per cell). Unlike DNA and RNA (represented as cDNA), currently there is no method for amplifying rare proteins in a cell extract. To obtain rare proteins

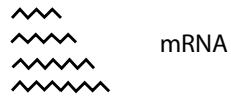
Fig. 1.11 (opposite page). Capturing genetic information in genomic or cDNA libraries. After extraction or conversion to DNA, DNA fragments are cloned and introduced into an appropriate host organism for propagation. Either DNA molecules or clones may be archived for future use. cDNA clones are usually small because mRNA from which cDNA is derived is usually hundreds of nucleotides to a few thousand nucleotides in length. Genomic clones may have small inserts (~ 2 kb, useful for DNA sequencing), intermediate-sized inserts (10 kb, useful for DNA sequence assembly), or large inserts (100–300 kb, useful for long-range sequence assembly). Molecules in the appropriate size range are produced by a size selection step prior to cloning. Different cloning vectors are required for different insert sizes.



Extract genomic DNA
from organism, organ,
or tissue

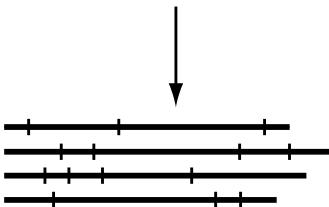
Extract mRNA
from organism, organ,
tissue, or tumor

- a. Incompletely digest with restriction endonuclease
- b. Purify one or more size classes (large, intermediate, or small)



mRNA

Reverse
transcription



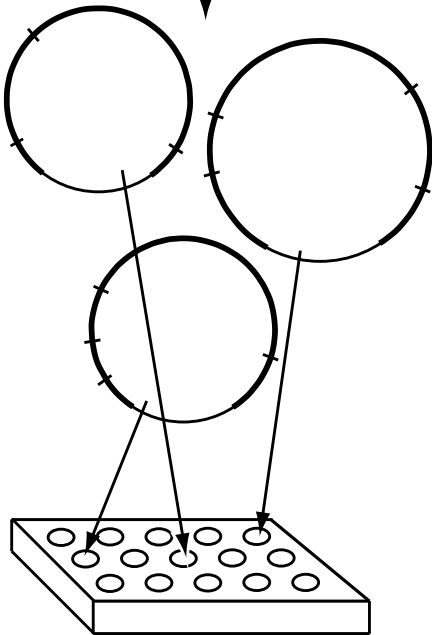
Duplex
genomic
DNA



Duplex
cDNA

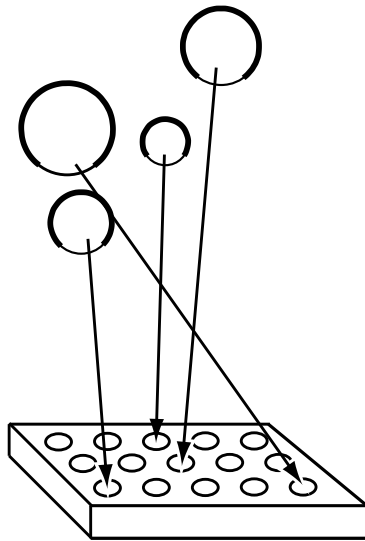
Clone

Clone



Genomic library

(large, intermediate, or small insert)



cDNA library

directly from cells, it may be necessary to employ many liters of cell culture and multiple purification steps to obtain even 10^{-3} g of purified protein. Large amounts of a particular protein can be obtained by cloning its coding sequence (possibly derived from a full-length cDNA) onto an **expression vector**—a vector with a promoter (constitutive or inducible) and sequences specifying the addition of polyA to transcripts. Large amounts of the desired protein are produced from such clones because of the increased gene copy number and increased transcription rate. However, a polypeptide chain produced in this manner may or may not be properly processed, folded, glycosylated, or assembled if it is expressed in a nonnative cell type.

If a particular protein has never before been purified, protein biochemists know a number of procedures to *try* (e.g., precipitation by $(\text{NH}_4)_2\text{SO}_4$, ion-exchange chromatography, high-pressure liquid chromatography, molecular sieve chromatography, etc.), but in general multiple steps are required, each of which must be optimized before milligram amounts of purified and active protein can be produced. During purification, proteins that function as parts of macromolecular complexes may be separated from other proteins in the complex and thus may lose biological activity. In addition, proteins **denature** (lose their natural structure) more readily than do nucleic acids. This can occur because of binding to surfaces (a problem particularly in dilute solutions), oxidation of sulfhydryl groups, unfolding at elevated temperatures, or exposure to detergents. (Nucleic acids are indifferent to detergents.) Similarly, some membrane proteins may be difficult to purify in soluble form because of their hydrophobic character.

A lack of amplification methods (short of expression cloning) and the wide range of protein properties determined by the various possible amino acid sequences influence methods for studying proteins. Two-dimensional polyacrylamide gel electrophoresis (**2DE**) is an established method for displaying (in principle) any cellular proteins in cell lysates or cell fractions (Fig. 1.12). Sample preparation disrupts the protein structures, producing a solution of denatured polypeptide chains. Therefore, functional proteins composed of multiple polypeptide chains (e.g., RNA polymerases) are “deconstructed” to their constituent polypeptides. Separation of the polypeptides on the polyacrylamide gel matrix depends upon two different properties associated with each polypeptide chain: the molecular mass and the isoelectric point (pI). Molecular mass is the summation of the molecular masses of the constituent amino acid residues, and since polypeptides generally differ in length and composition, each polypeptide species has a characteristic molecular mass. The isoelectric point of a protein or polypeptide is the pH at which its average charge is zero. The isoelectric point is related to the amino acid composition by the number and kind of acidic and basic residues that the polypeptide chain or protein contains. An excess of basic residues leads to a pI greater than 7.0, and an excess of acidic residues leads to a pI less than 7.0.

With 2DE (Fig. 1.12), the protein mixture is first resolved into a series of bands by **isoelectric focusing**, which is electrophoresis in a stationary

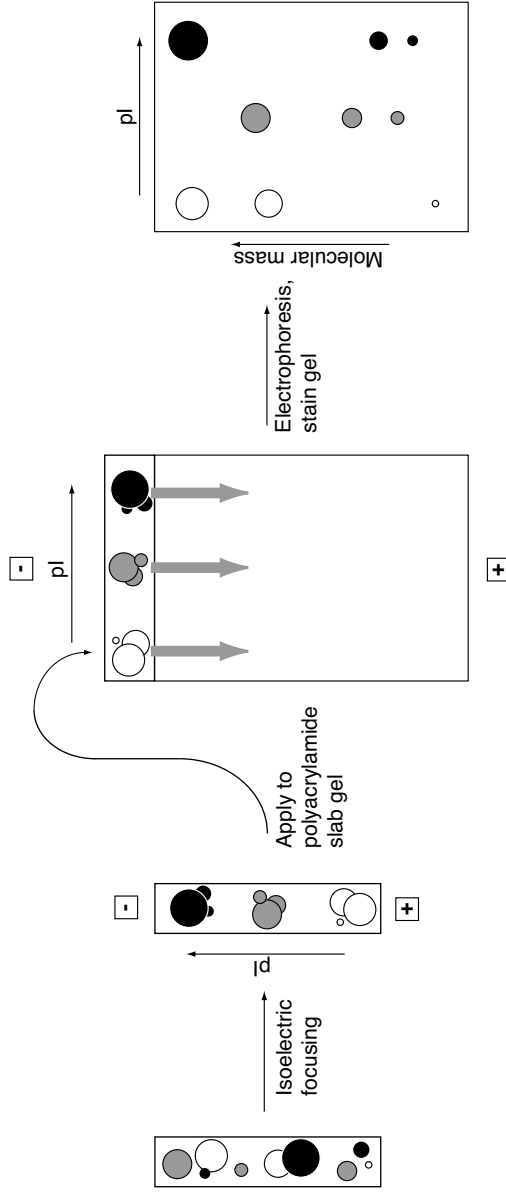


Fig. 1.12. Schematic illustration of two-dimensional gel electrophoresis. Circles represent proteins in the mixture, with different sizes representing different molecular masses. The shadings correspond to the isoelectric points of the proteins. Isoelectric focusing (left) separates the molecules into three different classes, each of whose members share the same pI. SDS-polyacrylamide gel electrophoresis (right) is conducted in an orthogonal direction, leading to the separation of proteins in each pI class into different protein spots.

pH gradient (i.e., a pH gradient for which the pH at any position is time-invariant). Proteins having an isoelectric point of 8, for example, migrate to that point in the pH gradient where the pH is 8 and then stop migrating because their average charge is zero at that point. Proteins whose isoelectric point is 4.5 migrate to the position in the pH gradient where the pH is 4.5. Isoelectric focusing is performed in a “tube gel” (typical tubes are 0.3 cm in diameter and 15 cm long) or on plastic-backed gel strips containing immobilized pH gradients. After the bands have been formed by isoelectric focusing, the gel or strip is equilibrated with a solution containing the strong detergent SDS. SDS is negatively charged. All polypeptides bind SDS at approximately the same mass of SDS/mass of protein, and they become extended and negatively charged with approximately the same charge-to-mass ratio in solution. The gel or strip is then placed on a slab gel, perpendicular to the direction of the electric field to be applied. Electrophoresis through the slab polyacrylamide gel resolves polypeptides based upon their extension in space, which is related to the molecular mass. After the electrophoresis step, spots corresponding to individual polypeptides can be visualized by staining or by autoradiography or phosphorimaging (if proteins were labeled with radioisotopes). Figure 1.13 shows an example of a stained gel. With typical protein loads, up to 1000 to 1500 polypeptides can be resolved in two dimensions.

Often, we wish to detect a specific macromolecule in the presence of others. For DNA and RNA, this is relatively easy because Watson-Crick base pairing is a specific and efficient mechanism for a probe DNA molecule to “recognize” a molecule containing the complementary sequence. However, there currently are no easy methods for detecting specific protein sequences, except for methods using antibodies and antibody-like molecules. These and similar methods are powerful and sensitive but are experimentally demanding, as described in the box below.

Antibodies and specific protein recognition

Antibodies (Ab) or **immunoglobulins** are proteins that are produced by vertebrate immune systems to bind “foreign” molecules that may be present in the body (e.g., bacteria). A complex combinatorial process produces antibodies that are capable of binding virtually any specific **antigen** (a molecule that elicits the immune response) that the organism might encounter. Usually (but not always), an antibody that recognizes and binds to antigen x will not recognize and bind to antigen y , and vice versa.

There are two labor-intensive steps in the production of antibodies: production of the antigen and production of the antibody. We have already discussed earlier the issues related to purification of proteins to be used as antigens. The second issue, antibody production, can be attacked in different ways. Traditionally, antibodies are made by repeated injection of an antigen into rabbits or goats and bleeding the animals several weeks later. This produces a sample of **polyclonal antibodies** (a *mixture* of different immunoglobu-

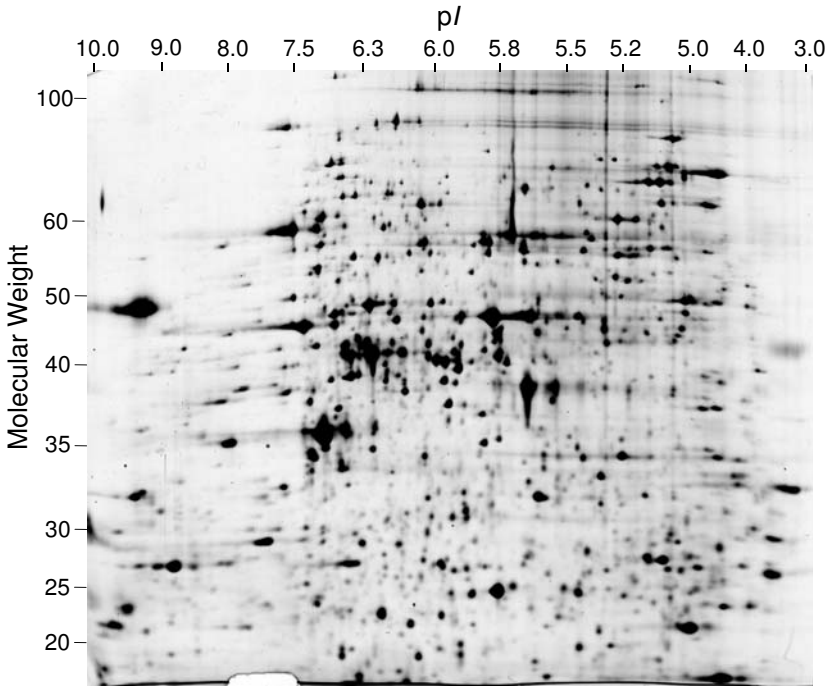


Fig. 1.13. Proteins extracted from yeast resolved by two-dimensional gel electrophoresis and visualized by silver staining. Molecular weights are in kilodaltons. Over 1000 spots were detected, corresponding to about 20% of all yeast genes. The abundance of proteins in the gel is approximately proportional to the spot intensity. Reprinted, with permission, from Gygi SP et al. (1999) *Molecular and Cell Biology* 19:1720–1730. Copyright 1999, the American Society for Microbiology. All rights reserved.

lin species directed against the antigen). The different immunoglobulins may “recognize” or bind to different specific regions of the antigen (different protein domains, for example). Different regions of the antigen recognized by distinct antibody species are called **epitopes**. A preferable but more expensive and time-consuming procedure is to produce a **monoclonal antibody**, which is a single immunoglobulin species that recognizes a single epitope. In this procedure, a mouse is immunized with the antigen, and cells from the spleen (where antibody-producing cells mature) are removed and fused with an “immortal” multiple myeloma (tumor) cell line. This produces hybrid cell lines (hybridomas), and individual hybridoma cell lines may produce a single antibody directed toward a single epitope from the original antigen. Such cells can be used to produce the antibody in tissue culture, or they can be injected into mice to produce the desired antibody *in vivo*. This procedure obviously

entails costs for vivarium facilities and for a hybridoma tissue culture facility. The procedure usually requires a few months.

An alternative to monoclonal antibodies are probe molecules identified by **phage display** approaches. With phage display, DNA encoding the antibody binding regions is cloned into one of the coat protein genes of a filamentous *E. coli* bacteriophage such as fd or M13. When the mature phage is produced, it “displays” the antigen binding region on its surface. From a mixture of bacteriophages that recognize a large number of antigens, a phage species capable of recognizing a particular antigen can be purified after repeated amplification and enrichment steps. This approach only requires standard cloning and biochemical expertise, but of course it still requires purified antigen. Production of antibodies can also be avoided by using small polypeptides specifically designed by protein engineering for specific protein binding, and such technology is commercially available (e.g., Affibody AB, Sweden).

1.5.3 Types of Experiments

In general, questions addressed by computational biology are subsets of the overall question, “How do cells and organisms function?” We may think of the larger question as being composed of three components:

- Characterizing the genome;
- Identifying patterns of gene expression and protein abundance under different conditions;
- Discovering mechanisms for controlling gene expression and the biochemical reactions in the cell.

We expand on these topics in the introductions to subsequent chapters, but here we provide an overview. Computational biologists should be concerned about experimental details because computational approaches must be tailored to the structure of the experimental data.

Genomes can be characterized by genetic and physical mapping, analyzing evolution of single-copy and repeated DNA sequences, identifying genes and their organization, and building inventories of genes by type. This is the realm of restriction mapping, cloning, DNA sequencing, pattern recognition, and phylogenetic tree building. Each of these topics is addressed in a subsequent chapter. Typically, DNA is isolated from an appropriate representative strain or lineage of a particular type of organism and cloned as shown in Fig. 1.11 (left). Clones stored in libraries may be digested with **restriction endonucleases** (individually or in combination) to produce maps of the various clones. Alternatively, the ends of the cloned inserts may be sequenced directly. Depending upon the purpose of the experiment, the clones may be screened by hybridization to find those having sequences similar to particular probe sequences (e.g., to DNA from a gene of interest). Ultimately the result

is the sequence of a DNA molecule annotated for any genes or regulatory signals (e.g., transcription factor binding sites) that it may contain. Comparison with similar gene regions in model organisms may provide insight into gene function. Investigators interested in those genes associated with a particular genetic disease may focus on a few genes, but with genome sequencing approaches, the entire panoply of genes is examined, usually in an evolutionary context.

Gene expression studies seek to measure the amounts of mRNA or protein species in a particular cell or tissue type under a particular set of physiological conditions. The **transcriptome** is the entire collection of transcripts, and the **proteome** is the entire collection of proteins for a particular cell and set of conditions. The transcriptome is studied by a variety of methods for measuring (directly or indirectly) mRNA levels, including **spotted microarray** experiments, “gene chip” experiments, serial analysis of gene expression (**SAGE**), and total gene expression analysis (**TOGA**). For eukaryotes, this may involve purification of RNA and preparing cDNA (Fig. 1.11, right). Each cDNA clone corresponds to a particular expressed sequence (mRNA). For spotted microarrays, gene or intergenic DNA samples spotted onto solid substrates are hybridized to labeled cDNA mixtures prepared from mRNA extracts (see Chapter 11). Proteomes can be analyzed by resolving protein extracts from cells by using 2DE and subjecting particular polypeptides to tandem mass spectrometry. Array technologies also are being devised to identify and quantify protein species in cell extracts.

Gene regulation may depend upon sites on DNA that bind regulatory proteins and also can depend upon protein-protein interactions. Sites on DNA that bind regulatory proteins can be identified on a DNA fragment by **gel-shift** or **footprinting** methods. Gel-shift experiments are lower-resolution electrophoretic assays that depend upon the retardation (“gel-shift”) of DNA-protein complexes relative to DNA having no bound protein. Alternatively, “footprinting” methods may be used to locate the position and extent of the binding region. These methods rely on reagents that cleave DNA within one or the other of the two strands. Proteins bound to the DNA protect it from cleavage. Fragmented DNA strands are resolved by gel electrophoresis, and the “footprint” is the region of the gel lacking cleaved fragments. Protein-DNA complexes formed *in vitro* can also be investigated by immunoprecipitation of complexes using antibodies specific for the bound protein.

Gene regulation can also depend upon protein-protein interactions, which can be studied *in vivo* by using yeast “two-hybrid” genetic systems. Protein-protein interactions can also be studied *in vitro* by chemical cross-linking. To detect proteins that interact with protein P, the extract containing it and possible binding partners is subjected to chemical cross-linking. Then reagents that specifically bind to P (a specific antibody, for example) are used to purify complexes containing P, and reversal of cross-linking releases proteins interacting with it. Such data are helpful for identifying the components and connectivity of protein interaction networks.

The particular combination of experiments employed will depend upon the reason for the study. The study may be part of an investigation of a particular genetic disease conducted in a medical school, or it may have been initiated within a biotechnology company to produce a profitable therapeutic agent. The study might be a comparison of a particular set of genes among a variety of organisms or might encompass the entire genome of a single organism. A wide range of methods derived from genetics, chemistry, biochemistry, and physics may be applied to each individual problem or project. Computational biologists should be aware of concepts associated with a wide range of disciplines.

References

- Alberts B, Lewis J, Raff M, Johnson A, Roberts K (2002) *Molecular Biology of the Cell* (4th edition). London: Taylor & Francis, Inc.
- Branden C, Tooze J (1998) *Introduction to Protein Structure* (2nd edition). London: Taylor & Francis, Inc.
- Calladine CR, Drew HR (1997) *Understanding DNA: The Molecule and How It Works* (2nd edition). San Diego, CA: Academic Press.
- Griffiths AJ, Lewontin RC, Gelbart WM, Miller JH, Gelbart W (2002) *Modern Genetic Analysis* (2nd edition). New York, NY: W. H. Freeman Company.
- Knoll AH, Carroll SB (1999) Early animal evolution: Emerging views from comparative biology and geology. *Science* 284:2129–2137.
- Mouse Genome Sequencing Consortium [MGSC] (2002). Initial sequencing and comparative analysis of the mouse genome. *Nature* 420:520–562.
- Pennisi E (2003) Modernizing the tree of life. *Science* 300:1692–1697.
- <http://www.ncbi.nlm.nih.gov/entrez/query.fcgi?db=Books>
Online books on genetics, molecular biology, and cell biology presented by the National Center for Biotechnology Information (NCBI). Includes some of the best standard textbooks.
- <http://www.ucmp.berkeley.edu/alllife/threedomains.html>
An excellent resource covering evolution and the diversity of life from both biological and paleontological perspectives. Maintained by the University of California, Berkeley Museum of Paleontology.
- <http://web.mit.edu/esgbio/www/7001main.html>
Biology hypertextbook. Information is highly compressed but rapidly accessible.
- <http://www.genome.gov/10002096>
Glossary of terms presented by the National Human Genome Research Institute. Includes illustrations for glossary entries.

Words

2.1 The Biological Problem

Consider the DNA sequence shown below:

```
TGATGATGAAGACATCAGCATTGAAGGGCTGATGGAACACATCCCGGGCCGGAC
TTCCCGACGGCGGCAATCATTAACGGTCGTCGCGGTATTGAAGAAGCTTACCGTA
CCGGTCGCGGCAAGGTGTATATCCGCGCTCGCGCAGAAGTGGAAGTTGACGCCAA
AACCGGTCGTGAAACCATTATCGTCCACGAAATCCGTATCAGGTAAACAAAGCG
CGCCTGATCGAGAAGATTGCGGAACTGGTAAAAGAAAAACGCGTGGAAGGCATCA
GCGCGCTGCGTGACGAGTCTGACAAAGACGGTATGCGCATCGTGATTGAAGTGAA
ACGCGATGCGGTGCGTGAAGTTGTGCTCAACAACCTCTACTCCAGACCCAGTTG
CAGGTTTCTTTCGGTATCAACATGGTGGCATTGCACCATGGTCAGCCGAAGATCA
TGAACCTGAAAGACATCATCGCGGCGTTTGTTCGTCACCGCCGTGAAGTGGTGAC
CCGTCGTAATAATTTTCGAACTGCGTAAAGCTCGCGATCGTGCTCATATCCTTGAA
GCATTAGCCGTGGCGCTGGCGAACATCGACCCGATCATCGAACTGATCCGTCATG
CGCCGACGCCCTGCAGAAGCGAAAACTGCGCTGGTTGCTAATCCGTGGCAGCTGGG
CAACGTTGCCGCGATGCTCGAACGTGCTGGCGACGATGCTGCGCGTCCGGAATGG
CTGGAGCCAGAGTTCGGCGTGCGTGATGGTCTGTACTACCTGACCGAACAGCAAG
CTCAGGCGATTCTGGATCTGCGTTTGCAGAACTGACCGGTCTTGAGCACGAAAA
ACTGCTCGACGAATACAAAGAGCTGCTGGATCAGATCGCGGAACTGTTGCGTATT
CTTGGTAGCGCGGATCGTCTGATGGAAGTATCCGTGAAGAGCTGGAGCTGGTTC
GTGAACAGTTCGGTGACAAACGTGCTACTGAAATCACCGCCAACAGCGCAGACAT
CAACCTGGAAGATCTGATACCCAGGAAGATGTGGTCTGACGCTCTCTACCCAG
GGCTACGTTAAGTATCAGCCGCTTCTGAATACGAAGCGCAGCGTCGTGGCGGGA
```

Given this sequence, there are a number of questions we might ask:

- What sort of statistics should be used to describe this sequence?
- Can we determine what sort of organism this sequence came from based on sequence content?
- Do parameters describing this sequence differ from those describing bulk DNA in that organism?

- What sort of sequence might this be: Protein coding? Centromere? Telomere? Transposable element? Control sequence?

This chapter approaches these sorts of questions from three different perspectives, all of which are united by considering **words**. These are short strings of letters drawn from an alphabet, which in the case of DNA is the set of letters **A**, **C**, **G**, and **T**. A word of length k is called a “ k -word” or “ k -tuple”; we use these terms interchangeably. For example, individual bases are 1-tuples, dinucleotides are 2-tuples, and codons are triplets, or 3-tuples. **GGGT** is a 4-word or 4-tuple.

DNA sequences from different sources or regions of a genome may be distinguished from each other by their k -tuple content. We begin by giving illustrations and examples of how word frequencies can vary within and between genomes. Second, we take a closer look at computational issues pertaining to words, starting with the seemingly obvious one about how to count words and how words can be located along a string. This includes describing their distribution in terms of a probabilistic model. Finally, we discuss various statistics that have been used to describe word frequencies.

2.2 Biological Words: $k = 1$ (Base Composition)

We consider first the frequencies of individual bases. For free-living organisms (in contrast with some bacteriophages and other viruses), DNA is typically duplex. This means that every **A** on one strand is matched by a **T** on the complementary strand, and every **G** on one strand is matched by **C** on the complementary strand. In other words, the number of **A** residues in the genome equals the number of **T** residues, and the number of **G** residues equals the number of **C** residues. This applies to the duplex DNA: as we will see later, the number of **G** or **A** residues on one strand need not equal the number of **C** or **T** residues (respectively) on the same strand. This statement is illustrated below:

5'-GGATCGAAGCTAAGGGCT-3'	Top strand:	7 G, 3 C
3'-CCTAGCTTCGATTCCCGA-5'	Duplex molecule:	10 G, 10 C

Considering **G** or **C** for this particular *duplex* DNA, it suffices to report the fraction $\text{fr}(\mathbf{G+C})$ of **G+C**, knowing the individual base frequencies will be $\text{fr}(\mathbf{G+C})/2$. Also, $\text{fr}(\mathbf{A+T}) = 1 - \text{fr}(\mathbf{G+C})$, so only a single parameter is required to describe the base frequencies for duplex DNA. (That is, there are four variables, $\text{fr}(\mathbf{A})$, $\text{fr}(\mathbf{C})$, $\text{fr}(\mathbf{G})$, and $\text{fr}(\mathbf{T})$, and there are three relations among them, $\text{fr}(\mathbf{A}) = \text{fr}(\mathbf{T})$, $\text{fr}(\mathbf{G}) = \text{fr}(\mathbf{C})$, and $\text{fr}(\mathbf{A+T}) = 1 - \text{fr}(\mathbf{G+C})$.)

Since the early days of molecular biology (before cloning and DNA sequencing), base composition has been used as a descriptive statistic for genomes of various organisms. The fraction $\text{fr}(\mathbf{G+C})$ of bulk DNA can be determined either by measuring the melting temperature of the DNA, T_m ,

or by determining the buoyant density of the DNA in a CsCl gradient using equilibrium ultracentrifugation. Both methods can reveal the presence of genome fractions having different base compositions, and in the case of ultracentrifuge measurements, bands of genomic DNA adjacent to the main band and differing from it in base composition may be described as “satellites.”

Table 2.1 presents the base compositions of DNA from a few organisms, indicating the range over which this statistic can vary. Obviously, there are constraints on base composition imposed by the genetic code: long homopolymers such as $\cdots \text{AAAAA} \cdots$ ($\text{fr}(\text{G}+\text{C}) = 0$) would not encode proteins having biochemical activities required for life. We see more about this when we consider $k = 3$ (codons).

Table 2.1. Base composition of various organisms. Bacterial data taken from the Comprehensive Microbial Resource maintained by TIGR: <http://www.tigr.org/tigr-scripts/CMR2/CMRHomePage.spl>.

Organism	%G+C	Genome size (Mb)
Eubacteria		
<i>Mycoplasma genitalium</i>	31.6	0.585
<i>Escherichia coli</i> K-12	50.7	4.693
<i>Pseudomonas aeruginosa</i> PAO1	66.4	6.264
Archaeobacteria		
<i>Pyrococcus abyssi</i>	44.6	1.765
<i>Thermoplasma volcanium</i>	39.9	1.585
Eukaryotes		
<i>Caenorhabditis elegans</i> (a nematode)	36	97
<i>Arabidopsis thaliana</i> (a flowering plant)	35	125
<i>Homo sapiens</i> (a bipedal tetrapod)	41	3,080

Another point will be visited in a later chapter: the distribution of individual bases within the DNA molecule is not ordinarily uniform. In prokaryotic genomes in particular, there is an excess of **G** over **C** on the **leading strands** (strands whose 5' to 3' direction corresponds to the direction of replication fork movement). This can be described by the “GC skew,” which is defined as $(\#G - \#C)/(\#G + \#C)$, calculated at successive positions along the DNA for intervals of specified width (“windows”); here, $\#G$ denotes the number of Gs and so on. As will be seen later, this quantity often changes sign at positions of replication origins and termini in prokaryotic genomic sequences. This is

another example of how a relatively simple statistic based on k -tuples with $k = 1$ can be informative.

In the sections that follow, we develop some probabilistic and statistical approaches for describing the base composition, dinucleotide composition, and other aspects of DNA sequences. To do this, it is most convenient to describe the DNA in terms of a single strand in a given 5' to 3' orientation. The other strand of the duplex is formed by taking its complement.

2.3 Introduction to Probability

This is a good point to introduce some necessary basic ideas of probability. We build on these ideas as we progress through this and later chapters. In this section, we use the nucleotide sequence on a single strand as an example. For definiteness, we assume that this sequence is written in a given 5' to 3' direction. We are often interested in deciding whether particular patterns of bases appear unusually often in a given sequence; such sequences might be of biological significance. To address such problems, we need a way to measure our “surprise” about the frequency of particular patterns, and to do this we use a probabilistic model for the sequence.

One way to specify such a probability model is to describe a method for simulating observations from the model. This means that we must specify the probabilistic rules the computer uses to produce the next letter in the simulated sequence, given the previous letters. We can then think of the sequence as the output of a machine (or simulation algorithm). Here is a simple set of rules that specify a probability model:

- (a) The first base in the sequence is either an A, C, G, or T with probability p_A, p_C, p_G, p_T , respectively.
- (b) Suppose the machine has generated the first r bases. To generate the base at position $r + 1$, the machine pays no attention to what has been generated before and spits out A, C, G, or T with the probabilities given in (a) above.

A run of the simulation algorithm results in a sequence of bases, and different runs will typically result in different sequences. The output of a random string of n bases will be denoted by L_1, L_2, \dots, L_n , where L_i denotes the base inserted in position i of the sequence. It is conventional to use small letters to denote the *particular* sequence that resulted from a run; we may observe $L_1 = l_1, L_2 = l_2, \dots, L_n = l_n$ for a particular simulation. In the next sections, we outline some basic probabilistic and statistical language that allows us to analyze such sequences.

2.3.1 Probability Distributions

Suppose that on a single step our machine produces an output X that takes exactly one of the J possible values in a set $\mathcal{X} = \{x_1, x_2, \dots, x_J\}$. In the DNA

sequence example, we have $J = 4$ and $\mathcal{X} = \{\mathbf{A}, \mathbf{C}, \mathbf{G}, \mathbf{T}\}$. We do not typically know with certainty which value in \mathcal{X} will be produced by our machine, so we call X a discrete **random variable**. (Note the font used to distinguish bases from random variables.) The term *discrete* refers to the fact that the set of possible values is finite. Now suppose that the value x_j occurs with probability p_j , $j = 1, 2, \dots, J$. We note that each p_j must be greater than or equal to 0, and the p_j must sum to 1; that is,

$$\begin{aligned} p_1, p_2, \dots, p_J &\geq 0; \\ p_1 + p_2 + \dots + p_J &= 1. \end{aligned}$$

We call the collection p_1, \dots, p_J the **probability distribution** of X , and we write

$$\mathbb{P}(X = x_j) = p_j, j = 1, 2, \dots, J.$$

In this book, we always use the symbol \mathbb{P} to denote probability. For example, the first base L_1 in our model for a DNA sequence has probability distribution

$$\mathbb{P}(L_1 = \mathbf{A}) = p_{\mathbf{A}}, \mathbb{P}(L_1 = \mathbf{C}) = p_{\mathbf{C}}, \mathbb{P}(L_1 = \mathbf{G}) = p_{\mathbf{G}}, \mathbb{P}(L_1 = \mathbf{T}) = p_{\mathbf{T}}. \quad (2.1)$$

Note that some textbooks use the term *probability mass function* of the random variable instead of *probability distribution*, defined above. The probability distribution allows us to compute probabilities of different outcomes in the following way. If S is an event (that is, a subset of \mathcal{X}), then the probability that S occurs, written $\mathbb{P}(X \in S)$, is calculated as

$$\mathbb{P}(X \in S) = \sum_{j: x_j \in S} p_j.$$

The term $j : x_j \in S$ is read “ j such that x_j is in S .” For example, if $S = \{\mathbf{G}, \mathbf{C}\}$, then $\mathbb{P}(X \in S) = p_{\mathbf{G}} + p_{\mathbf{C}}$.

In the following sections, we study the probability distribution of the number of times a given pattern occurs in a random DNA sequence L_1, L_2, \dots, L_n , and we’ll make our patterns one base long to begin with. To address this question, we define a new sequence X_1, X_2, \dots, X_n by

$$X_i = \begin{cases} 1, & \text{if } L_i = \mathbf{A}, \\ 0, & \text{otherwise.} \end{cases} \quad (2.2)$$

The number of times N that \mathbf{A} appears is then the sum of the X s:

$$N = X_1 + X_2 + \dots + X_n. \quad (2.3)$$

Starting from the probability distribution (2.1) of the L_i , we can find the probability distribution of each of the X_i as follows:

$$\begin{aligned} \mathbb{P}(X_i = 1) &= \mathbb{P}(L_i = \mathbf{A}) = p_{\mathbf{A}}, \\ \mathbb{P}(X_i = 0) &= \mathbb{P}(L_i = \mathbf{C} \text{ or } \mathbf{G} \text{ or } \mathbf{T}) = p_{\mathbf{C}} + p_{\mathbf{G}} + p_{\mathbf{T}} = 1 - p_{\mathbf{A}}. \end{aligned} \quad (2.4)$$

Different “runs” of our machine produce strings having different values of N . We ultimately wish to know what a “typical” value of N might be, which means we need to know its probability distribution. To find the probability distribution of N is more complicated because we need to know how the individual outputs from our machine are related to each other. This is the topic of the next section.

2.3.2 Independence

According to our simple DNA sequence model, the probability distribution of the base in position $r + 1$ does not depend on the bases occupying positions $r, \dots, 2, 1$. This captures the notion that outputs from the machine do not influence each other (you might like to ponder whether this is likely to be true in a DNA sequence). In this section, we formalize the notion of independence for a collection of discrete random variables X_1, X_2, \dots, X_n . Capturing this notion in a definition is a little complicated.

Discrete random variables X_1, X_2, \dots, X_n are said to be **independent** if, for $r = 2, 3, \dots, n$,

$$\begin{aligned} \mathbb{P}(X_{i_1} = a_1, X_{i_2} = a_2, \dots, X_{i_r} = a_r) = \\ \mathbb{P}(X_{i_1} = a_1)\mathbb{P}(X_{i_2} = a_2) \cdots \mathbb{P}(X_{i_r} = a_r) \end{aligned}$$

for all subsets $\{i_1, i_2, \dots, i_r\}$ of $\{1, 2, \dots, n\}$ and for all possible values a_1, \dots, a_r . In particular, if X_1, \dots, X_n are independent, we can calculate the probability of a set of outcomes by using the *multiplication rule for probabilities of independent events*: for events $A_i, i = 1, 2, \dots, n$, we have

$$\mathbb{P}(X_1 \in A_1, \dots, X_n \in A_n) = \mathbb{P}(X_1 \in A_1)\mathbb{P}(X_2 \in A_2) \cdots \mathbb{P}(X_n \in A_n). \quad (2.5)$$

For the DNA sequence model outlined in the introduction, the L_i are indeed independent, and so the probability of obtaining the sequence l_1, l_2, \dots, l_n is given by the product of the probabilities of each l_i ,

$$\mathbb{P}(L_1 = l_1, \dots, L_n = l_n) = \mathbb{P}(L_1 = l_1)\mathbb{P}(L_2 = l_2) \cdots \mathbb{P}(L_n = l_n), \quad (2.6)$$

where the terms on the right-hand side are determined by the probability distribution of a single base given in (2.1).

In the last section, we introduced a sequence of discrete random variables X_1, \dots, X_n that counted whether the bases in the sequence L_1, L_2, \dots, L_n were A or not. It is intuitively clear that if the L_i are independent of one another, then so too are the X_i defined in (2.2). You should check this from the definition. While it is sometimes possible to check whether a collection of random variables X_1, X_2, \dots, X_n are independent, it is more common to *assume* independence and then use the multiplication rule (2.5) to calculate probabilities of different outcomes. Random variables that are independent and have the same probability distribution are said to be *independent and identically distributed*; in what follows, we use the abbreviation “iid.” For further discussion of *dependent* random variables, see Exercise 8.

2.3.3 Expected Values and Variances

In this section we describe measures of location (or central tendency) and spread for random variables that take on numerical values. Suppose that X is a discrete random variable taking values in \mathcal{X} , a subset of $(-\infty, \infty)$. We define the **expected value** (or *mean* or *expectation*) of X by

$$\mathbb{E}X = \sum_{j=1}^J x_j \mathbb{P}(X = x_j) = x_1 p_1 + x_2 p_2 + \cdots + x_J p_J. \quad (2.7)$$

In this book, the symbol \mathbb{E} will always be used to indicate expected value or mean. For the random variables X_i defined in (2.2) with distribution given in (2.4), we have

$$\mathbb{E}X_i = 1 \times p_A + 0 \times (1 - p_A) = p_A. \quad (2.8)$$

If we know the expected value of the random variable X , then it is easy to calculate the expected random variable $Y = cX$ for any constant c ; we obtain

$$\mathbb{E}Y = c \mathbb{E}X.$$

The random variable N in (2.3) counts the number of times the base **A** appears in the sequence L_1, L_2, \dots, L_n . We do not yet know the probability distribution of N , but we can compute its expected value in another way. We use the fact that the mean of the sum of the X s is the sum of the means of the X s. That is, for any random variables X_1, X_2, \dots, X_n , we have

$$\mathbb{E}(X_1 + X_2 + \cdots + X_n) = \mathbb{E}X_1 + \mathbb{E}X_2 + \cdots + \mathbb{E}X_n. \quad (2.9)$$

It follows from this and the result in (2.8) that the expected number of times we see an **A** in our n bp sequence is

$$\mathbb{E}N = \mathbb{E}X_1 + \mathbb{E}X_2 + \cdots + \mathbb{E}X_n = n \mathbb{E}X_1 = np_A. \quad (2.10)$$

The expected value of a random variable X gives a measure of its location; values of X tend to be scattered around this value. In addition to this measure of location, we need a measure of spread: Is X closely concentrated about its expected value, or is it spread out? To measure spread, we use a quantity called the **variance**. We define the variance of X by

$$\text{Var}X = \mathbb{E}(X - \mu)^2 = \sum_{j=1}^J (x_j - \mu)^2 p_j, \quad (2.11)$$

where $\mu = \mathbb{E}X$ is defined in (2.7). It can be shown that

$$\text{Var}X = \mathbb{E}X^2 - \mu^2 = \sum_{j=1}^J x_j^2 p_j - \mu^2, \quad (2.12)$$

a formula that sometimes simplifies calculations. For the random variables X_i in (2.4), we see that

$$\text{Var}X_i = [1^2 \times p_A + 0^2 \times (1 - p_A)] - p_A^2 = p_A(1 - p_A). \quad (2.13)$$

The (positive) square root of the variance of X is called its **standard deviation** and is denoted by $\text{sd}(X)$. If X is multiplied by the constant c , then the variance is multiplied by c^2 ; that is, if $Y = cX$

$$\text{Var}Y = \text{Var}(cX) = c^2 \text{Var}(X).$$

The standard deviation of Y is then given by

$$\text{sd}Y = \text{sd}(cX) = |c| \text{sd}(X).$$

To calculate the variance of the number N of **A**s in our DNA sequence, we exploit the fact that *the variance of a sum of independent random variables is the sum of the individual variances*; that is, if X_1, \dots, X_n are independent random variables,

$$\text{Var}(X_1 + \dots + X_n) = \text{Var}(X_1) + \dots + \text{Var}(X_n). \quad (2.14)$$

When this rule is applied to N , we find from (2.13) that

$$\text{Var}N = n \text{Var}X_1 = np_A(1 - p_A). \quad (2.15)$$

Equations (2.14) and (2.15) give two statistics that describe features of the probability distribution of N mentioned at the end of Section 2.3.1.

2.3.4 The Binomial Distribution

The expected value and variance of a random variable such as N are just two (of many) summary statistics that describe features of its probability distribution. Much more information is provided by the probability distribution itself, which we show how to calculate in this section.

To do this, notice that $\mathbb{P}(X_1 = x_1, \dots, X_n = x_n)$ is the same for any x_1, x_2, \dots, x_n containing the same number of 1s. Furthermore, the fact that the X_i are iid means that we can use (2.5) to see that if there are j 1s in x_1, x_2, \dots, x_n , then the probability of that sequence is $p^j(1 - p)^{n-j}$, where, for typographical convenience, we have written $p = p_A$. Finally, to compute the probability that the sequence contains j **A**s (i.e., that $N = j$), we need to know how many different realizations of the sequence x_1, x_2, \dots, x_n have j 1s (and $n - j$ 0s). This is given by the binomial coefficient $\binom{n}{j}$, defined by

$$\binom{n}{j} = \frac{n!}{j!(n-j)!}, \quad (2.16)$$

where $j! = j(j-1)(j-2)\cdots 3\cdot 2\cdot 1$, and by convention $0! = 1$. It now follows that the probability of observing j A s is

$$\mathbb{P}(N = j) = \binom{n}{j} p^j (1-p)^{n-j}, j = 0, 1, 2, \dots, n. \quad (2.17)$$

The probability distribution in (2.17) is known as the **binomial distribution** with parameters n (the number of trials) and p (the probability of success). The mean of N , which can also be calculated using (2.17) and (2.7), is given in (2.10), and the variance of N is given in (2.15).

2.4 Simulating from Probability Distributions

To understand the behavior of random variables such as N , it is useful to simulate a number of instances having the same probability distribution as N . If we could get our computer to output numbers N_1, N_2, \dots, N_n having the same distribution as N , we could use them to study the properties of this distribution. For example, we can use the sample mean

$$\bar{N} = (N_1 + N_2 + \cdots + N_n)/n \quad (2.18)$$

to estimate the expected value μ of N . We could use the sample variance

$$s^2 = \frac{1}{n-1} \sum_{i=1}^n (N_i - \bar{N})^2 \quad (2.19)$$

to estimate the variance σ^2 of N , and we can use a histogram of the values of N_1, \dots, N_n to estimate the probability of different outcomes for N .

To produce such a string of observations, we need to tell the computer how to proceed. We need the computer to be able to generate a series of random numbers that we can use to produce N_1, \dots, N_n . This is achieved by use of what are called *pseudo-random numbers*. Many programming languages (and the statistical environment R we use in this book is no exception) have available algorithms (or random number generators) that generate sequences of random numbers U_1, U_2, \dots that behave as though they are independent and identically distributed, have values in the unit interval $(0, 1)$, and satisfy, for any interval (a, b) contained in $(0, 1)$,

$$\mathbb{P}(a < U_1 \leq b) = b - a.$$

Random variables with this property are said to be *uniformly distributed on* $(0, 1)$. This last phrase captures formally what one understands intuitively: any number between 0 and 1 is a possible outcome, and each is equally likely. Uniform random numbers form the basis of our simulation methods. From now on, we assume we have access to as many such U s as we need.

To illustrate the use of our random number generator, we will simulate an observation with the distribution of X_1 in (2.2). We can do this by taking a uniform random number U and setting $X_1 = 1$ if $U \leq p \equiv p_A$ and 0 otherwise. This works because the chance that $U \leq p$ is just p . Repeating this procedure n times (with a new U each time) results in a sequence X_1, X_2, \dots, X_n from which N can be computed by adding up the X s.

We can use a similar approach to simulate the sequence of bases L_1, L_2, \dots, L_n . This time we divide up the interval (0,1) into four intervals with endpoints at $p_A, p_A + p_C, p_A + p_C + p_G$, and $p_A + p_C + p_G + p_T = 1$. If the simulated U lies in the leftmost interval, set $L_1 = A$; if it is in the second interval, set $L_1 = C$; if it is in the third interval, set $L_1 = G$; and otherwise set $L_1 = T$. Repeating this procedure with a new value of U each time will produce a sequence of bases L_1, L_2, \dots, L_n . Fortunately, we do not have to write code to implement this approach, as it is included in R already as the `sample` function. The details are given in the box below.

Computational Example 2.1: Simulating a DNA sequence

The `sample` function can be used to generate many sorts of samples. The function has the form

```
sample(x,n,replace=TRUE,pi)
# x = list of values to be sampled
# n = number of samples required
# replace=TRUE means sampling with replacement
# pi = probability distribution for the list in x
```

Here is an application that generates ten outcomes from the probability distribution in (2.4) with $p_A = 0.25$:

```
> pi<-c(0.25,0.75)
> x<-c(1,0)
> sample(x,10,replace=TRUE,pi)
[1] 1 0 0 0 0 0 1 1 1 0
```

We can use a similar approach to generate a DNA sequence according to our simple iid model. First, we code the bases as $A = 1$, $C = 2$, $G = 3$, and $T = 4$ and assume that each base is equally likely. To simulate 10,000 bases under this model and look at the first 15 bases, we can use

```
> pi<-c(0.25,0.25,0.25,0.25)
> x<-c(1,2,3,4)
> seq<-sample(x,10000,replace=TRUE,pi)
> seq[1:15]
[1] 4 4 4 4 1 1 2 3 2 4 2 2 1 1 1
```

It is sometimes convenient to be able to use the same string of random numbers more than once (for example, when preparing examples for this book!). To do this, you can use

```
set.seed(int)
```

where `int` is an integer seed. Try generating two DNA sequences without using `set.seed()` and then by calling `set.seed(100)` before each run. Compare the outputs!

By looking through a given simulated sequence, we can count the number of times a particular pattern arises (for example, the one-letter pattern A considered earlier) and so, by repeatedly generating sequences and analyzing each of them, we can get a feel for whether or not our particular pattern is “unusual.” We illustrate this by simulating observations having the binomial distribution with $p = 0.25$ and $n = 1000$. Recall that under our uniform base frequency model for DNA, this is the distribution of the number of A s in the sequence of length n . R can perform binomial simulations, as described in the box below.

Computational Example 2.2: Simulating binomial random variables

R has a number of built-in functions for simulating observations from standard distributions. To generate 2000 observations from a binomial distribution with $n = 1000$ trials and success probability $p = 0.25$, we can use

```
> x <- rbinom(2000,1000,0.25)
```

The sample mean (see (2.18)) of our simulated values can be found using

```
> mean(x)
[1] 249.704
```

This value is in good agreement with the mean of N , which is $\mu = np = 250$. The variance of the replicates (see (2.19)) can be found using the square of the sample standard deviation:

```
> sd(x)^2
[1] 183.9734
```

Once more, this is in good agreement with the theoretical value of $\sigma^2 = np(1-p) = 187.5$. To plot the histogram, we use

```
> hist(x,xlab="Number of successes",main="")
```

The result is shown in Fig. 2.1.

Later in the book, a number of statistical concepts are introduced by use of **simulation**. For now, we answer another question about the frequency of the pattern A in a sequence. Suppose then that we have a sequence of length 1000 bp and assume that each base is equally likely (and so has probability 0.25). How likely is it to observe at least 280 A s in such a sequence? There

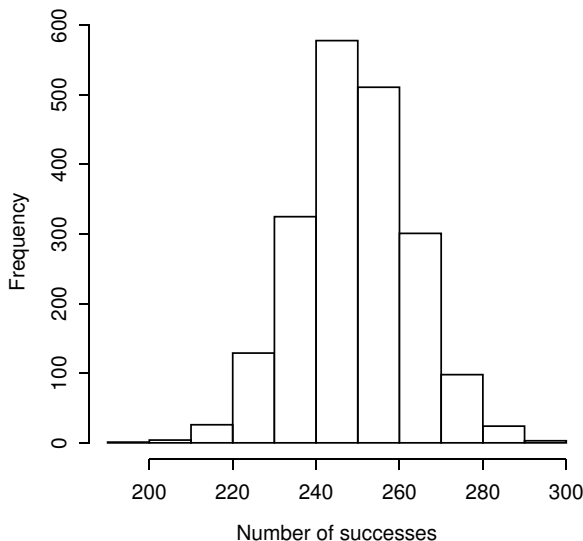


Fig. 2.1. Histogram of 2000 replicates of a binomial random variable having $n = 1000$ trials and success probability $p = 0.25$.

are three ways to attack this problem: by using the distribution in (2.17), by simulation, and by an approximation known as the Central Limit Theorem. We discuss the first two approaches here and the third in Section 3.4.

For the first approach, the probability we need to calculate is given by

$$\mathbb{P}(N \geq 280) = \sum_{j=280}^{1000} \binom{1000}{j} (1/4)^j (1 - 1/4)^{1000-j}. \quad (2.20)$$

A computer algebra program such as DERIVETM gives the answer 0.01644. The second approach is to simulate a large number of random variables having the distribution of N and calculate how many times the values are greater than or equal to 280. In 10,000 runs of this procedure (see Exercise 4), 149 values were at least 280, so the estimate of the required probability is $149/10,000 \approx 0.015$, in good agreement with the theoretical value of ≈ 0.016 .

2.5 Biological Words: $k = 2$

If l_i is a nucleotide at position i , then a dinucleotide is $l_i l_{i+1}$ (5' to 3' polarity implied). Since l_i is drawn from an alphabet of four bases A, C, G, T, there are 16 different dinucleotides: AA, AC, AG, ..., TG, GG. Since the sum of

the dinucleotide frequencies is 1, just 15 of them suffice to give a complete description of the dinucleotide frequencies in a single-stranded molecule. Dinucleotides are important in part because physical parameters associated with them can describe the trajectory of the DNA helix through space (DNA bending), which may have effects on gene expression. Here we concentrate only on their abundances.

Now suppose we model the sequence L_1, L_2, \dots, L_n using our iid model with base probabilities given by (2.1). Since the bases are behaving independently of one another, we can use the multiplication rule (2.5) for probabilities to calculate the probabilities of each dinucleotide $r_1 r_2$ as

$$\mathbb{P}(L_i = r_1, L_{i+1} = r_2) = p_{r_1} p_{r_2}. \quad (2.21)$$

For example, under the independence model, the chance of seeing the dinucleotide AA is p_A^2 , and the chance of seeing CG is $p_C p_G$.

To see whether a given sequence has unusual dinucleotide frequencies compared with the iid model, we compare the observed number O of the dinucleotide $r_1 r_2$ with the expected number given by $E = (n - 1)p_{r_1} p_{r_2}$. (Note that $n - 1$ is the number of dinucleotides in a string of length n .) One statistic to use is

$$X^2 = \frac{(O - E)^2}{E}. \quad (2.22)$$

The rationale for using this statistic is as follows. If the observed number is close to the expected number (so that the model is doing a good job of predicting the dinucleotide frequencies), X^2 will be small. If, on the other hand, the model is doing a poor job of predicting the dinucleotide frequencies, then X^2 will tend to be large.

All that remains is to determine which values of X^2 are unlikely if in fact the model is true. This turns out to be a subtle problem that is beyond the scope of this book, but we can at least give a recipe. For further details, see Exercise 10.

(a) Calculate the number c given by

$$c = \begin{cases} 1 + 2p_{r_1} - 3p_{r_1}^2, & \text{if } r_1 = r_2; \\ 1 - 3p_{r_1} p_{r_2}, & \text{if } r_1 \neq r_2. \end{cases}$$

(b) Calculate the ratio X^2/c , where X^2 is given in (2.22).

(c) If this ratio is larger than 3.84, conclude that the iid model is not a good fit.

If the base frequencies are unknown, the same approach works if the frequencies $\text{fr}(\text{A})$, $\text{fr}(\text{C})$, $\text{fr}(\text{G})$, and $\text{fr}(\text{T})$ are estimated from the data. Table 2.2 presents the observed values of X^2/c for the first 1000 bp of two organisms, *E. coli* (GenBank ID NC_000913) and *Mycoplasma genitalium* (GenBank ID L43967). It can be seen that *E. coli* dinucleotide frequencies are not well-described by the simple iid model, whereas the *M. genitalium* sequence is not

as bad. As one might have expected, given the biological nature of genomic sequences, there are some dinucleotides whose frequencies differ significantly from what would be expected from the iid model.

Table 2.2. Observed values of X^2/c for the first 1000 bp of each genome. For *E. coli*, the base frequencies were taken as (0.25, 0.25, 0.25, 0.25), whereas for *M. genitalium* they were (0.45, 0.09, 0.09, 0.37), close to the observed frequencies. Significant values are in bold.

Dinucleotide	Observed X^2/c for	
	<i>E. coli</i>	<i>M. genitalium</i>
AA	6.78	0.15
AC	0.05	1.20
AG	5.99	0.18
AT	0.01	0.01
CA	2.64	0.01
CC	0.03	0.39
CG	0.85	4.70
CT	4.70	1.10
GA	2.15	0.34
GC	10.04	1.07
GG	0.01	0.09
GT	1.76	0.61
TA	5.99	1.93
TC	9.06	2.28
TG	3.63	0.05
TT	1.12	0.13

2.6 Introduction to Markov Chains

As we can see from Table 2.2, real genomes have sequence properties more complicated than those described by our simple iid model. A more complicated probabilistic model is required to capture the dinucleotide properties of real genomes. One approach is to use a **Markov chain**, a natural generalization of a sequence of independent trials. Many applications of Markov chains in computational biology are described in Durbin et al. (1998). Suppose that we examine a sequence of letters corresponding to the genome of an organism. If we focus on position n , we realize that the character at that position might be dependent upon the letters preceding it. For example, human DNA has a lower than expected frequency of the dinucleotide 5'-CG-3': if we have a C at position $t-1$, then the probability of a G at position t will be lower than might be expected if the letter at position $t-1$ were A, G, or T. To make these ideas

precise, we make use of more ideas from probability, particularly the notion of *conditional probability*.

2.6.1 Conditional Probability

We consider events, which are subsets of the sample space Ω . In the earlier examples, events were usually defined in terms of outcomes of random variables, so that Ω corresponds to the set \mathcal{X} of possible outcomes of a single experiment. In particular, $\mathbb{P}(\Omega) = 1$, and

$$\mathbb{P}(A) + \mathbb{P}(A^c) = 1,$$

for any event A , where A^c denotes the complement $\Omega - A$ of A . For two events A and B , we define the intersection of A and B , written $A \cap B$, as the set of elements in Ω belonging to both A and B . The union of A and B , written $A \cup B$, is the set of elements of Ω belonging to either A or B (and possibly both). The conditional probability of A given B , denoted by $\mathbb{P}(A \mid B)$, is defined by

$$\mathbb{P}(A \mid B) = \frac{\mathbb{P}(A \cap B)}{\mathbb{P}(B)}, \quad (2.23)$$

when $\mathbb{P}(B) > 0$ (and, by convention, $= 0$ if $\mathbb{P}(B) = 0$). The term $\mathbb{P}(A \cap B)$ is read “probability of A and B .”

A number of useful consequences derive from this definition, among them Bayes’ Theorem, which states that

$$\mathbb{P}(B \mid A) = \frac{\mathbb{P}(A \mid B)\mathbb{P}(B)}{\mathbb{P}(A)}. \quad (2.24)$$

Suppose next that B_1, B_2, \dots, B_k form a *partition* of Ω :

- (a) The B_i are disjoint (i.e., $B_i \cap B_j = \emptyset$ for $i \neq j$)
- (b) and exhaustive (i.e., $B_1 \cup B_2 \cup \dots \cup B_k = \Omega$).

Another useful identity is known as the law of total probability: for any event A , and a partition B_1, \dots, B_k ,

$$\begin{aligned} \mathbb{P}(A) &= \sum_{i=1}^k \mathbb{P}(A \cap B_i) \\ &= \sum_{i=1}^k \mathbb{P}(A \mid B_i)\mathbb{P}(B_i). \end{aligned} \quad (2.25)$$

A number of applications of these results are given in the exercises at the end of the chapter.

2.6.2 The Markov Property

We study a sequence of random variables $\{X_t, t = 0, 1, 2, \dots\}$ taking values in the state space \mathcal{X} . For example, X_t might be the letter in position t of a DNA sequence, and the state space is the set $\mathcal{X} = \{\mathbf{A}, \mathbf{C}, \mathbf{G}, \mathbf{T}\}$. The sequence $\{X_t, t \geq 0\}$ is called a *first-order Markov chain* if the probability of finding a particular character at position $t + 1$ given the preceding characters at positions $t, t - 1$, and so forth down to position 0 is identical to the probability of observing the character at position $t + 1$ given the character state of the immediately preceding position, t . In other words, only the previous neighbor influences the probability distribution of the character at any position. More formally, $\{X_t, t \geq 0\}$ is called a first-order Markov chain if it satisfies the *Markov property*,

$$\mathbb{P}(X_{t+1} = j \mid X_t = i, X_{t-1} = i_{t-1}, \dots, X_0 = i_0) = \mathbb{P}(X_{t+1} = j \mid X_t = i),$$

for $t \geq 0$ and for all $i, j, i_{t-1}, \dots, i_0 \in \mathcal{X}$. Markov chains of order k correspond to the case where the conditional distribution of the present position depends on the previous k positions. We do not consider higher-order Markov chains in this book.

We consider Markov chains that are homogeneous, which means the probability $\mathbb{P}(X_{t+1} = j \mid X_t = i)$ is independent of the position t in the sequence. For example, $\mathbb{P}(X_{t+1} = \mathbf{G} \mid X_t = \mathbf{C})$ is the same throughout the sequence if the Markov chain is homogeneous. The probabilities common to all positions are denoted by p_{ij} ,

$$p_{ij} = \mathbb{P}(X_{t+1} = j \mid X_t = i), \quad i, j \in \mathcal{X}.$$

The p_{ij} are the elements of a matrix P called the *one-step transition matrix* of the chain. In the matrix P below, we show what the transition matrix would look like for DNA. Each row corresponds to one of the possible states at position t (i.e., row 1 corresponds to $X_t = \mathbf{A}$), and each column corresponds to the possible state at $t + 1$ ($X_{t+1} = \mathbf{A}, \mathbf{C}, \mathbf{G}$, or \mathbf{T}):

$$P = \begin{pmatrix} p_{AA} & p_{AC} & p_{AG} & p_{AT} \\ p_{CA} & p_{CC} & p_{CG} & p_{CT} \\ p_{GA} & p_{GC} & p_{GG} & p_{GT} \\ p_{TA} & p_{TC} & p_{TG} & p_{TT} \end{pmatrix}.$$

As we indicated, if the Markov chain is homogeneous, then this transition matrix applies all along the chain. Since after any position there must be one of the characters from \mathcal{X} , we see that $\sum_j p_{ij} = 1$ for each value of i . If all the rows in P were identical, then the next position from any starting position would have the same distribution, regardless of the identity of the character at the current position. This corresponds to the iid model we used in Section 2.3.2.

The transition matrix tells us the probabilities that apply as we go from the state at position t to the state at position $t+1$. For example, if $X_t = \mathbf{A}$, then the probability that $X_{t+1} = \mathbf{G}$ is $p_{\mathbf{AG}}$. But how do we start the chain? To completely specify the evolution of the Markov chain, we need both the transition matrix and the *initial probability distribution*, π . The initial probability distribution can be written as a row vector whose elements are

$$\pi_i = \mathbb{P}(X_0 = i), i \in \mathcal{X}.$$

In the case of DNA, the π_i are the initial probabilities (at position 0) of A, C, G, and T.

To find the probability distribution for the states at position 1 (represented by row vector $\pi^{(1)}$), we use the law of total probability as follows:

$$\begin{aligned} \mathbb{P}(X_1 = j) &= \sum_{i \in \mathcal{X}} \mathbb{P}(X_0 = i, X_1 = j) \\ &= \sum_{i \in \mathcal{X}} \mathbb{P}(X_0 = i) \mathbb{P}(X_1 = j | X_0 = i) \\ &= \sum_{i \in \mathcal{X}} \pi_i p_{ij}. \end{aligned} \tag{2.26}$$

You may recognize this as the product of the row vector π with the matrix P , so that (in matrix notation)

$$\pi^{(1)} = \pi P.$$

To compute the probability distribution for the states at position 2 (represented by row vector $\pi^{(2)}$), we first note that $\mathbb{P}(X_2 = j | X_0 = i)$ is the i, j th element of the matrix $PP = P^2$. This is another application of the law of total probability,

$$\begin{aligned} \mathbb{P}(X_2 = j | X_0 = i) &= \sum_{k \in \mathcal{X}} \mathbb{P}(X_2 = j, X_1 = k | X_0 = i) \\ &= \sum_{k \in \mathcal{X}} \mathbb{P}(X_2 = j | X_1 = k, X_0 = i) \mathbb{P}(X_1 = k | X_0 = i) \\ &= \sum_{k \in \mathcal{X}} \mathbb{P}(X_2 = j | X_1 = k) \mathbb{P}(X_1 = k | X_0 = i) \\ &= \sum_{k \in \mathcal{X}} p_{ik} p_{kj} \\ &= (PP)_{ij}, \end{aligned}$$

as required. (Note that the second line follows from the definition of conditional probability, the third from the Markov property.) Copying the argument that leads to (2.26) then shows that the elements of $\pi^{(2)}$ are given by

$$\pi_j^{(2)} = \mathbb{P}(X_2 = j) = \sum_i \pi_i (P^2)_{ij}.$$

This can be generalized to the t th position, giving

$$\mathbb{P}(X_t = j) = \sum_i \pi_i (P^t)_{ij},$$

where the elements $(P^t)_{ij}$ correspond to the elements of the matrix generated by multiplying the transition matrix by itself t times (a total of t factors). This expression gives the probability distribution, $\mathbb{P}(X_t = j)$, at any position t .

It is possible that the distribution $\pi^{(t)}$ is the same for every t . This is called a **stationary distribution** of the chain. This occurs when

$$\pi_j = \sum_i \pi_i p_{ij} \text{ for all } j.$$

In matrix notation, this condition is $\pi = \pi P$. If X_0 also has π as its distribution, then $\pi = \pi P^t$ and

$$\mathbb{P}(X_t = j) = \pi_j.$$

This shows that X_t then has the same distribution for every t . Note that this does not contradict the dependence of the state at t on the state at $t - 1$.

2.6.3 A Markov Chain Simulation

To illustrate the use of a Markov chain, we use the observed dinucleotide frequencies of *M. genitalium* to determine the parameters of a Markov model. The observed dinucleotide relative frequencies are given below; each row specifies a base, and each column specifies the following base:

$$\begin{array}{c} \text{A} \quad \text{C} \quad \text{G} \quad \text{T} \\ \text{A} \left(\begin{array}{cccc} 0.146 & 0.052 & 0.058 & 0.089 \\ 0.063 & 0.029 & 0.010 & 0.056 \\ 0.050 & 0.030 & 0.028 & 0.051 \\ 0.087 & 0.047 & 0.063 & 0.140 \end{array} \right). \end{array} \quad (2.27)$$

The individual base frequencies (the base composition) may be calculated from the matrix by summing across the rows. We obtain $p_A = 0.345$, $p_C = 0.158$, $p_G = 0.159$, and $p_T = 0.337$. To estimate the AA element of the transition matrix P of the chain, we note that

$$p_{AA} = \mathbb{P}(X_t = \text{A} \mid X_{t-1} = \text{A}) = \frac{\mathbb{P}(X_t = \text{A}, X_{t-1} = \text{A})}{\mathbb{P}(X_{t-1} = \text{A})} \approx \frac{0.146}{0.345} = 0.423.$$

In this calculation, we used the observed dinucleotide frequency matrix to find the proportion 0.146 of the dinucleotide AA and the base frequency vector to find the proportion 0.345 of A. The same estimation procedure can be done

for the 15 other entries of P , and we arrive at an estimated transition matrix of

$$P = \begin{array}{c} \text{A} \\ \text{C} \\ \text{G} \\ \text{T} \end{array} \begin{pmatrix} \text{A} & \text{C} & \text{G} & \text{T} \\ 0.423 & 0.151 & 0.168 & 0.258 \\ 0.399 & 0.184 & 0.063 & 0.354 \\ 0.314 & 0.189 & 0.176 & 0.321 \\ 0.258 & 0.138 & 0.187 & 0.415 \end{pmatrix}.$$

The rows sum to unity, to within rounding error, as they should. The smallest matrix element ($p_{CG} = 0.063$) corresponds to placing **G** at a position given **C** at the previous position. For our initial distribution π , we assign vector elements just using the base composition:

$$\pi = (0.345, 0.158, 0.159, 0.337).$$

Now we are ready to simulate a sequence string that resembles *M. genitalium* DNA, at least in terms of dinucleotide frequencies. We don't expect it to look like actual *M. genitalium* DNA because our probabilistic model is still likely to be too simple: the model is capable of generating the correct proportions of k -tuples with $k = 1$ and $k = 2$, but it does not include the "machinery" for simulating k -tuple frequencies for $k > 2$. The details are given in Computational Example 2.3.

Computational Example 2.3: Simulating a string having characteristics of *Mycoplasma* DNA

We generate a sequence having 50,000 positions. We code the bases as follows: **A** = 1, **C** = 2, **G** = 3, and **T** = 4. By using numbers instead of characters, we can use logical operators (`==`, `!=`, `>`, ...) in our analysis of the sequence. The values for the transition matrix and π were presented above. We simulate the sequence with the aid of R (Appendix A). First, we write an R function (program) to generate the sequence. For this function, we supply the following input variables (arguments): the transition matrix P , π , and n , the length of the string to be generated. We also supply a vector x containing the characters to be sampled. A function that will simulate the sequence is presented below:

```
> markov1 <- function(x,pi,P,n){
  # x = vector [1 2 3 4] representing A, C, G, T, respectively
  # pi = the probability distribution for X0: (1x4 row vector)
  # P = transition matrix (4x4)
  # n = length of simulated sequence
  # Initialize vector to contain simulated sequence
  mg <- rep(0,n)
  # Produce initial element
  mg[1] <- sample(x,1,replace=TRUE,pi)
  for(k in 1:(n-1)){
```



```

mg[k+1]<-sample(x,1,replace=T,P[mg[k],])
      }
return(mg)
}

```

Lines prefixed by # are comments and are not executed. The R library function `sample()` is employed to generate an element to be placed at position $i + 1$ given a particular letter at i . Use the `help(sample)` command at the R prompt for documentation for this function. Note particularly how the probability distributions `pi` and `P[i,]`, the rows of the transition matrix, are employed for each use of `sample()`. Each application of `markov1` will produce a different string (check this), but the overall properties of each string should be similar. To input the parameters in the simulation, we use:

```

> x <- c(1:4) # Loading parameters
> pi <- c(.342,.158,.158,.342)
> P <- matrix(scan(),ncol=4, nrow=4,byrow=T)
1: .423 .151 .168 .258
5: .399 .184 .063 .354
9: .314 .189 .176 .321
13: .258 .138 .187 .415
17: # enter "return" here to end input

```

Application of `markov1` uses the following:

```

> tmp<-markov1(x,pi,P,50000)

```

Checking the simulation output

We can check the base composition (remembering that C is represented by 2 and G is represented by 3) of the generated sequence:

```

> length(tmp[tmp[]==1])
[1] 16697
> length(tmp[tmp[]==2])
[1] 8000
> length(tmp[tmp[]==3])
[1] 7905
> length(tmp[tmp[]==4])
[1] 17398
> (8000+7905)/(16697+8000+7905+17398) # compute fr(G+C)
[1] 0.3181

```

This compares favorably with the value 31.6% G+C given in the transition matrix. Now we check whether `tmp` contains an appropriate fraction of CG dinucleotides:

```

> count=0
> for(i in 1:49999){ # 49999 because i+1 undefined for 50000
+ if(tmp[i]==2 && tmp[i+1]==3)
+ count<-count+1}
> count
[1] 482
> count/49999
[1] 0.0096

```

From (2.27), the relative abundance of the CG dinucleotide in *M. genitalium* is 0.010, whereas the string produced by the Markov model contains CG at a relative abundance 0.0096. This matches the observed data well. You should verify that other dinucleotide relative abundances are correctly predicted by your simulation. Evidently, the Markov model provides a good probabilistic description of the data for *M. genitalium* DNA.

2.7 Biological Words with $k = 3$: Codons

As mentioned in Chapter 1, there are 61 codons that specify amino acids and three stop codons. Since there are 20 common amino acids, this means that most amino acids are specified by more than one codon. This has led to the use of a number of statistics to summarize the “bias” in codon usage. An example of such a statistic is shown later. To show how these codon frequencies can vary, consider the specific example of the *E. coli* proteins. Table 2.3 displays the predicted and observed codon relative frequencies for three (out of 20) particular amino acids found in 780 genes of *E. coli*. (At the time this work was done, no complete genome sequences were available for any organism.) The predicted relative frequencies are calculated as follows.

For a sequence of independent bases L_1, L_2, \dots, L_n the expected 3-tuple relative frequencies can be found by using the logic employed for dinucleotides in (2.21). We calculate the probability of a 3-word as

$$\begin{aligned} \mathbb{P}(L_i = r_1, L_{i+1} = r_2, L_{i+2} = r_3) = \\ \mathbb{P}(L_i = r_1)\mathbb{P}(L_{i+1} = r_2)\mathbb{P}(L_{i+2} = r_3). \end{aligned} \quad (2.28)$$

This provides the expected frequencies of particular codons. To get the entries in Table 2.3, we calculate the relative proportion of each of the codons making up a particular amino acid. Using the base frequencies from Table 2.1, we find that

$$\mathbb{P}(\text{TTT}) = 0.246 \times 0.246 \times 0.246 = 0.01489,$$

while

$$\mathbb{P}(\text{TTC}) = 0.246 \times 0.246 \times 0.254 = 0.01537.$$

It follows that among those codons making up the amino acid Phe, the expected proportion of TTT is

$$\frac{0.01489}{0.01489 + 0.01537} = 0.492.$$

Allowing for approximations in the base frequencies of *E. coli*, this is the value given in the first row of the second column in Table 2.3.

Table 2.3. Comparison of predicted and observed triplet frequencies in coding sequences for a subset of genes and codons from *E. coli*. Figures in parentheses below each gene class show the number of genes in that class. Data were taken from Médigue et al. (1991).

Codon Predicted		Observed		
		Gene Class I (502)	Gene Class II (191)	
Phe	TTT	0.493	0.551	0.291
	TTC	0.507	0.449	0.709
Ala	GCT	0.246	0.145	0.275
	GCC	0.254	0.276	0.164
	GCA	0.246	0.196	0.240
	GCG	0.254	0.382	0.323
Asn	AAT	0.493	0.409	0.172
	AAC	0.507	0.591	0.828

Médigue et al. (1991) clustered the different genes into three groups based on such codon usage patterns, and they observed three clusters. For Phe and Asn different usage patterns are observed for Gene Class I and Gene Class II. For Gene Class II in particular, the observed codon frequencies differ considerably from their predicted frequencies. When Médigue et al. checked the gene annotations (names and functions), they found that Class II genes were largely those such as ribosomal proteins or translation factors—genes expressed at high levels—whereas Class I genes were mostly those that are expressed at moderate levels.

A statistic that can describe each protein-coding gene for any given organism is the **codon adaptation index**, or CAI (Sharp and Li, 1987). This statistic compares the distribution of codons actually used in a particular protein with the preferred codons for highly expressed genes. (One might also compare them to the preferred codons based on gene predictions for the whole genome, but the CAI was devised prior to the availability of whole-genome sequences.) Consider a sequence of amino acids $X = x_1, x_2, \dots, x_L$ representing protein X , with x_k representing the amino acid residue corresponding to codon k in the gene. We are interested in comparing the actual codon usage

with an alternative model: that the codons employed are the most probable codons for highly expressed genes. For the codon corresponding to a particular amino acid at position k in protein X , let p_k be the probability that *this* particular codon is used to code for the amino acid in highly expressed genes, and let q_k correspond to the probability for *the most frequently used* codon of the corresponding amino acid in highly expressed genes. The CAI is defined as

$$\text{CAI} = \left[\prod_{k=1}^L p_k/q_k \right]^{1/L}.$$

In other words, the CAI is the geometric mean of the ratios of the probabilities for the codons *actually* used to the probabilities of the codons *most frequently* used in highly expressed genes. An alternative way of writing this is

$$\log(\text{CAI}) = \frac{1}{L} \sum_{k=1}^L \log(p_k/q_k).$$

This expression is in terms of a sum of the logarithms of probability ratios, a form that is encountered repeatedly in later contexts.

For an example of how this works, consider the amino acid sequence from the amino terminal end of the *himA* gene of *E. coli* (which codes for one of the two subunits of the protein IHF: length $L = 99$). This is shown in Fig. 2.2, and below it are written the codons that appear in the corresponding gene. Underneath is a table showing probabilities (top half) and corresponding codons (in corresponding order) in the bottom half. The maximum probabilities (the q_k) are underlined. The CAI for this fragment of coding sequence is then given by

$$\text{CAI} = \left[\frac{1.000}{1.000} \times \frac{0.199}{\underline{0.469}} \times \frac{0.038}{\underline{0.888}} \times \frac{0.035}{\underline{0.468}} \cdots \right]^{1/99}.$$

The numerator of each fraction corresponds to p_k , the probability that the *observed* codon in the *himA* gene sequence would actually be used in a highly expressed gene. If every codon in a gene corresponded to the most frequently used codon in highly expressed genes, then the CAI would be 1.0. In *E. coli*, a sample of 500 protein-coding genes displayed CAI values in the range from 0.2 to 0.85 (Whittam, 1996).

Why do we care about statistics such as the CAI? As we will see in Chapter 11, there is a correlation between the CAI and mRNA levels. In other words, the CAI for a gene sequence in genomic DNA provides a first approximation of its expression level: if the CAI is relatively large, then we would predict that the expression level is also large.

If we wanted a probabilistic model for a genome, $k = 3$, we could employ a second (or higher)-order Markov chain. In the second-order model, the state at position $t + 1$ depends upon the states at both t and $t - 1$. In this case, the transition matrix could be represented by using 16 rows (corresponding to all

M	A	L	T	K	A	E	M	S	E	Y	L	F	...
ATG	GCG	CTT	ACA	AAA	GCT	GAA	ATG	TCA	GAA	TAT	CTG	TTT	...
<u>1.000</u>	<u>0.469</u>	0.018	0.451	<u>0.798</u>	<u>0.469</u>	<u>0.794</u>	<u>1.000</u>	<u>0.428</u>	<u>0.794</u>	<u>0.193</u>	0.018	<u>0.228</u>	
	0.057	0.018	<u>0.468</u>	0.202	0.057	0.206		0.319	0.206	<u>0.807</u>	0.018	<u>0.772</u>	
	0.275	0.038	0.035		0.275			0.033			0.038		
	0.199	0.033	0.046		0.199			0.007			0.033		
		0.007						0.037			0.007		
		<u>0.888</u>						0.176			<u>0.888</u>		
ATG	GCT	TTA	ACT	AAA	GCT	GAA	ATG	TCT	GAA	TAT	TTA	TTT	
	GCC	TTG	ACC	AAG	GCC	GAG		TCC	GAG	TAC	TTG	TTC	
	GCA	CTT	ACA		GCA			TCA			CTT		
	GCG	CTC	ACG		GCG			TCG			CTC		
		CTA						AGT			CTA		
		CTG						AGC			CTG		

Fig. 2.2. Example of codon usage patterns in *E. coli* for computation of the codon adaptation index of a gene. The probability for the most frequently used codon in highly expressed genes is underlined.

possible dinucleotide states for $t - 1$ and t) and four columns (corresponding to the possible states at position $t + 1$). We do not explore this further here.

2.8 Larger Words

The number and distributions of k -tuples, $k > 3$, can have practical and biological significance. Some particularly important k -tuples correspond to $k = 4, 5, 6$, or 8 . These include recognition sites for restriction endonucleases (e.g., 5'-AGCT-3' is the recognition sequence for endonuclease *A*luI, 5'-GAATTC-3' is the recognition sequence for *E*coRI, and 5'-GCGGCCGC-3' is the recognition sequence for *N*otI). The distribution of these k -tuples throughout the genome will determine the distribution of restriction endonuclease digest fragments ("restriction fragments"). These distributions are discussed in Chapter 3. There are also particular words (e.g., Chi sequences 5'-GCTGGTGG-3' in *E. coli*, $k = 8$) that may be significantly over-represented in particular genomes or on one or the other strands of the genome. For example, Chi appears 761 times in the *E. coli* chromosome compared with approximately 70 instances predicted using base frequencies under the iid model. Moreover, Chi sequences are more abundant on the leading strand than on the **lagging strand**. These observations relate in part to the involvement of Chi sequences in generalized recombination. Another example is the uptake sequences that function in bacterial transformation (e.g., 5'-GCCGTCTGAA-3' in *Neisseria gonorrhoeae*, $k = 10$). Other examples of over-represented k -tuples can be found in the review by Karlin et al. (1998). Some sequences may be under-represented. For example, 5'-CATG-3' occurs in the *E. coli* K-12 chromosome at about 1/20 of the expected frequency.

k -words ($k \geq 4$) are also useful for analyzing particular genomic subsequences. At the end of the next chapter, we illustrate how 4-word frequencies can be used to quantify the differences between *E. coli* promoter sequences and “average” genomic DNA.

2.9 Summary and Applications

In the cases $k = 1, 2$, and 3 above, we saw that frequencies of words or statistics derived from them (GC skew for $k = 1$) were not as predicted from the independent, identically distributed base model. This is no surprise: genomes code for biological information, and we would therefore not expect the iid model to provide an accurate description for real genomes. The frequencies of k -tuples have a number of applications. We already mentioned that GC skew can be used to predict locations of replication origins and termini in prokaryotes. Prokaryotes also may engage in gene transfer, and local genome regions having aberrant base compositions may indicate genome segments acquired by lateral transfer. For eukaryotes, gene regions may have on average a different base composition than regions outside genes (e.g., human genes are relatively GC-rich compared with the genome as a whole).

For $k = 3$, we saw that different gene classes have different codon usage frequencies. In general, the distribution of codon usage differs from organism to organism. The codon usage pattern of an anonymous DNA sequence from an organism can be compared against the overall usage pattern for that organism to help determine whether the reading frame being analyzed probably is, or is not, a protein-coding region. In Sections 2.5 and 2.7, words were described in terms of probabilistic models. Sometimes the observed frequencies of k -words can be used to make inferences about DNA sequences. For example, suppose that we were given a sequence string that hypothetically could be a portion of a candidate exon or prokaryotic gene:

GACGTTAGCTAGGCTTTAATCCGACTAAACCTTTGATGCATGCCTAGGCTG

Simply by noting the stop codons (underlined) in all three reading frames, and knowing that a typical bacterial gene contains, on average, more than 300 codons, or that the typical human exon, for example, contains around 50 codons, we can make a reasonable inference that this string does not code for a protein.

k -tuple frequencies can assist in classifying DNA sequences by content, such as predicting whether an unannotated sequence is coding or noncoding. Because coding sequences commonly specify amino acid strings that are functionally constrained, we would expect that their distribution of k -tuple frequencies would differ from that of noncoding sequences (e.g., intergenic or intronic sequences). Consider in-frame hexamers ($k = 6$). There are 4096 of these 6-tuple words. We can already predict from known polypeptide sequence

data that some 6-tuples will be infrequent. For example, the pair of residues Trp-Trp in succession is not common in protein sequences, which implies that the corresponding dicodon hexamer, TTGTTG, is likely to be relatively infrequent in coding sequences. Alternatively, we could use $k = 3$ and compute the CAI within open reading frames to identify those that might correspond to highly expressed genes (i.e., CAI close to 1.0). k -tuple frequencies and other content measures such as the presence of particular signals (see Chapter 9) are among the statistical properties employed by computational gene-finding tools.

References

- Campbell AM, Mrázek J, Karlin S (1999) Genome signature comparisons among prokaryote, plasmid, and mitochondrial DNA. *Proceedings of the National Academy of Sciences USA* 96:9184–9189.
- DERIVE™ 5. See <http://education.ti.com/us/product/software/derive/features/features.html>
- Durbin R, Eddy S, Krogh A, Mitchison G (1998) *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids*. Cambridge: Cambridge University Press.
- Karlin S, Campbell AM, Mrázek J (1998) Comparative DNA analysis across diverse genomes. *Annual Review of Genetics* 32:185–225.
- Médigue C, Rouxel T, Vigier P, Hénaut A, Danchin A (1991) Evidence for horizontal gene transfer in *Escherichia coli* speciation. *Journal of Molecular Biology* 222:851–856.
- Sharp PM, Li W-H (1987) The codon adaptation index—a measure of directional synonymous codon usage bias, and its potential applications. *Nucleic Acids Research* 15:1281–1295.
- Whittam TS (1996) In Neidhardt FC (Ed. in Chief), Curtiss III R, Ingraham JL, Lin ECC, Low KB, Magasanik B, Reznikoff WS, Riley M, Schaechter M, Umberger HE (eds). *Escherichia coli and Salmonella: Cellular and Molecular Biology*. Washington, D.C.: ASM Press, pp 2708–2720.

Exercises

Exercise 1. The base composition of a certain microbial genome is $p_G = p_C = 0.3$ and $p_A = p_T = 0.2$. We are interested in 2-words where the letters are assumed to be independent. There are $4 \times 4 = 16$ 2-words.

- Present these 16 probabilities in a table. (Do your 16 numbers sum to 1.0?)
- Purine bases are defined by $R = \{A, G\}$ and pyrimidine bases by $Y = \{C, T\}$. Let E be the event that the first letter is a pyrimidine, and F the event

that the second letter is A or C or T. Find $\mathbb{P}(E)$, $\mathbb{P}(F)$, $\mathbb{P}(E \cup F)$, $\mathbb{P}(E \cap F)$, and $\mathbb{P}(F^c)$.

(c) Set $G = \{\text{CA}, \text{CC}\}$. Calculate $\mathbb{P}(G | E)$, $\mathbb{P}(F | G \cup E)$, $\mathbb{P}(F \cup G | E)$.

Exercise 2. For three events A , B , and C show that $\mathbb{P}(A \cap B | C) = \mathbb{P}(A | B \cap C)\mathbb{P}(B | C)$.

Exercise 3. The independent random variables X and Y have the following expectations: $\mathbb{E}(X) = 3$, $\mathbb{E}(X^2) = 12$, $\mathbb{E}(Y) = 5$, and $\mathbb{E}(Y^2) = 30$. Find

(a) $\mathbb{E}(X + Y)$, $\mathbb{E}(2X + 1)$, $\mathbb{E}(2X + 0.3Y)$ and $\mathbb{E}(2X - 0.3Y)$.

(b) $\text{Var}X$, $\text{Var}Y$, $\text{Var}(2X + 5)$, $\text{Var}(X + Y)$, $\text{Var}(5X + 7Y)$, $\text{Var}(5X - 7Y)$, and $\text{Var}(5X + 7Y + 1600)$.

Exercise 4. Suppose N has a binomial distribution with $n = 10$ and $p = 0.3$.

(a) Using the formula (2.17), calculate $\mathbb{P}(N = 0)$, $\mathbb{P}(N = 2)$, $\mathbb{E}(N)$, and $\text{Var}N$.

(b) Using R and Computational Example 2.2, simulate observations from N .

Use the simulated values to estimate the probabilities you calculated in (a), and compare with the results in (a).

(c) Now use R to simulate observations from N when $n = 1000$ and $p = 0.25$.

What is your estimate of $\mathbb{P}(N \geq 280)$? (See (2.20).)

Exercise 5. Verify the terms in the first row of the transition matrix P presented in Section 2.6.3. Describe how you would use the sequence of *M. genitalium* to produce this matrix.

Exercise 6. Find the stationary distribution of the chain with transition matrix P in Section 2.6.3; that is, solve the equations $\pi = \pi P$ subject to the elements of π begin positive and summing to 1. Compare π to the base composition of *M. genitalium*, and comment.

Exercise 7. Using the values for P in Section 2.6.3, compute P^2 , P^4 and P^8 . (Remember to use `%%` as the matrix multiplication operator in R, not `*`.) What quantities are the row entries approaching? This distribution is called the *limiting distribution* of the chain. Compare to the results of Exercise 6.

Exercise 8. Perform the simulation in Chapter 2.6.3, and verify that the appropriate dinucleotide frequencies are produced in your simulated string.

Exercise 9. Using the sequence of *E. coli* (GenBank ID NC_000913) and the method in Section 2.6.3, find the dinucleotide frequencies and the estimated transition matrix. (Hint: Download the sequence in FASTA format from the NCBI website, and convert letters to numbers using a text editor.)

Exercise 10. In this example, we use R to verify the distribution of the statistic X^2 given in (2.22), as used in Table 2.2. To do this, first choose a pair of bases $r_1 r_2$, and calculate the appropriate value of c by following the recipe after (2.22) for the given base frequencies $p = (p_1, \dots, p_4)$. Now use R to

repeatedly simulate strings of 1000 letters having distribution p , calculate O (the number of times the pair of letters $r_1 r_2$ is observed) and E , and hence X^2/c . Plot a histogram of these values, and compare it to the theoretical distribution (which is the χ^2 distribution with 1 degree of freedom). Remark: This simulation approach provides a useful way to estimate percentage points of the distribution of *any* test statistic.

Exercise 11. The genome composition π of *E. coli* can be computed from Table 2.1. Take the first 1000 bps of the *E. coli* sequence you used in the previous exercise. We are going to use a variant of (2.22) to test if this 1000 bp sequence has an unusual base composition when compared with π . The statistic to use is

$$X^2 = \sum_{i=1}^4 \frac{(O_i - E_i)^2}{E_i}, \quad (2.29)$$

where O_i denotes the number of times base i is observed in the sequence, and E_i denotes the expected number (assuming that frequencies are given by π).

- Calculate O_i and E_i , $i = 1, \dots, 4$, and then X^2 .
- Values of X^2 that correspond to unusual base frequencies are determined by the large values of the χ^2 distribution with $4 - 1 = 3$ degrees of freedom. Using a 5% level of significance, are data consistent with π or not? [Hint: percentage points of the χ^2 distribution can be found using R.]

Exercise 12. In this exercise we have two random variables X and Y which are not independent. Their joint probability distribution is given in the following table:

		Y			
		1	3	6	9
X	2	0.11	0.05	0.20	0.08
	3	0.20	0.02	0.00	0.10
	7	0.00	0.05	0.10	0.09

The values of X are written in the first column and the values of Y in the first row. The table is read as $\mathbb{P}(X = 7 \& Y = 6) = 0.10$, and so on.

- Find the marginal distribution of X and Y . (That is, $\mathbb{P}(X = 2), \mathbb{P}(X = 3), \dots$)
- Write $Z = XY$. Find the probability distribution of Z .
- The **covariance** between any two random variables is defined by

$$\text{Cov}(X, Y) = \mathbb{E}(X - \mathbb{E}X)(Y - \mathbb{E}Y).$$

Show that $\text{Cov}(X, Y) = \mathbb{E}(XY) - \mathbb{E}X \times \mathbb{E}Y$.

- Find $\mathbb{E}X, \mathbb{E}Y, \sigma_X^2 = \text{Var}X, \sigma_Y^2 = \text{Var}Y$, and $\text{Cov}(X, Y)$ for the example in the table.

- (d) The **correlation coefficient** ρ is defined by $\rho_{X,Y} = \text{Cov}(X,Y)/\sigma_X\sigma_Y$. It can be shown that $-1 \leq \rho \leq 1$, the values ± 1 arising when Y is a linear function of X . Verify this last statement.
- (e) Calculate ρ for the example in the table.

Exercise 13. Using R, simulate n pairs of observations $(X_i, Y_i), i = 1, 2, \dots, n$ from the distribution in the table in Exercise 12.

- (a) From these observations calculate the estimates \bar{X}, \bar{Y}, s_X^2 , and s_Y^2 (see (2.18), (2.19)).
- (b) Calculate the estimate $s_{X,Y}^2$ of $\text{Cov}(X, Y)$ defined by

$$s_{X,Y}^2 = \frac{1}{n-1} \sum_{i=1}^n (X_i - \bar{X})(Y_i - \bar{Y}).$$

- (c) Calculate the estimate r of the correlation coefficient ρ via $r = s_{X,Y}^2 / (s_X s_Y)$.
- (d) Compare the estimated covariance and correlation obtained for different values of n with the theoretical values obtained in Exercise 12.

Word Distributions and Occurrences

3.1 The Biological Problem

Suppose that we wanted to obtain a sample of DNA that contained a specific gene or portion of a gene with very little other DNA. How could we do this? Today, given a genome sequence, we could design PCR primers flanking the DNA of interest and by PCR could amplify just that segment. Prior to the development of rapid genomic sequencing technologies, the process was much more complicated.

DNA is a macromolecule. That means that DNA molecules can have very high molecular weights. Because DNA can be long (for example, the DNA in human chromosome 1, at 225,000,000 bp, is 7.65 cm long) but is only 20×10^{-8} cm thick, it is easily broken by hydrodynamic shear. Such a long molecule cannot be transferred from one tube to another without breakage during pipetting. This was found to be a problem even with smaller molecules (e.g., 50,000 bp). The result of shearing is a collection of DNA fragments that are not broken at the same position, so that molecules containing the gene of interest intact might be very rare. What would be desirable is a method for cutting out the DNA at reproducible locations on the larger molecule together with a method for amplifying this DNA segment. Restriction endonucleases provided the means for precisely and reproducibly cutting the DNA into fragments of manageable size (usually in the size range of 100s to 1000s of base pairs), and molecular cloning provided the method for amplifying the DNA of interest (Section 1.5.1). The clone containing the DNA of interest could be identified by hybridization of a probe DNA (known to contain the sequence of interest) to DNA from bacterial colonies (if the DNA had been cloned into plasmid vectors) or from plaques (if a bacteriophage cloning vector had been used).

The cloned DNA fragment can be put in context with other fragments (which themselves can be subsequently analyzed) by creating a type of physical map called a restriction map. A **restriction map** is a display of positions on a DNA molecule of locations of cleavage by one or more restriction en-

endonucleases. It is created by determining the ordering of the DNA fragments generated after digestion with one or more restriction endonucleases. The restriction map is useful not only for dissecting a DNA segment for further analysis but also as a “fingerprint” or bar code that distinguishes that molecule from any other molecule. Even a list of fragments and their sizes can serve as a kind of fingerprint. Given a sufficiently large collection of mapped clones, it is possible to build up a restriction map of the DNA from which the clones were derived by matching restriction site patterns at the ends of the inserts. Again, remember that cloning puts DNA of manageable size into vectors that allow the inserted DNA to be amplified, and the reason for doing this is that large molecules cannot be readily manipulated without breakage.

The overall process can be summarized as shown in Figure 3.1: given a DNA molecule, digest the DNA with one or more restriction endonucleases to generate the particular set of fragments dictated by the sequence of that molecule; determine the sizes of the product fragments by acrylamide or agarose gel electrophoresis; and then, using one or more techniques, infer the locations of sites (or equivalently, the order of fragments) in the original DNA molecule.

When the Human Genome Project was originally contemplated, it was supposed that a physical map would first be constructed so that appropriate restriction fragments (of known location) could then be sequenced. In the strictest “top-down” approach, we would first construct a high-resolution genetic map, then clone the DNA into large insert vectors such as yeast artificial chromosomes (YACs) or bacterial artificial chromosomes (BACs), subclone the inserts of these into cosmids, which would then be fingerprinted after restriction endonuclease digestion, and then subclone the cosmid inserts into plasmids for sequencing (Section 8.4.3). The location of any sequence on the genome would then be determined by tracing back from plasmid to cosmid to YAC, taking into account the restriction map of inserts at each step. This did not take into account the powerful computational tools that would later become available, which then made possible random, shotgun sequencing approaches (sequencing randomly chosen small insert clones, followed by computational sequence assembly). Restriction enzyme digests are still employed in the shotgun method to assess the contents of the collection of BACs, but prior physical mapping is not required.

Although restriction mapping is not as central as it once was for genome analysis, workers at the bench still use restriction mapping to evaluate the content of clones or DNA constructs of interest, so it is still important to talk about locations and distributions of restriction endonuclease recognition sites. This chapter presents the probabilistic basis for analyzing this kind of problem. In addition, the last section shows how word occurrences can be used to characterize biologically significant DNA subsequences.

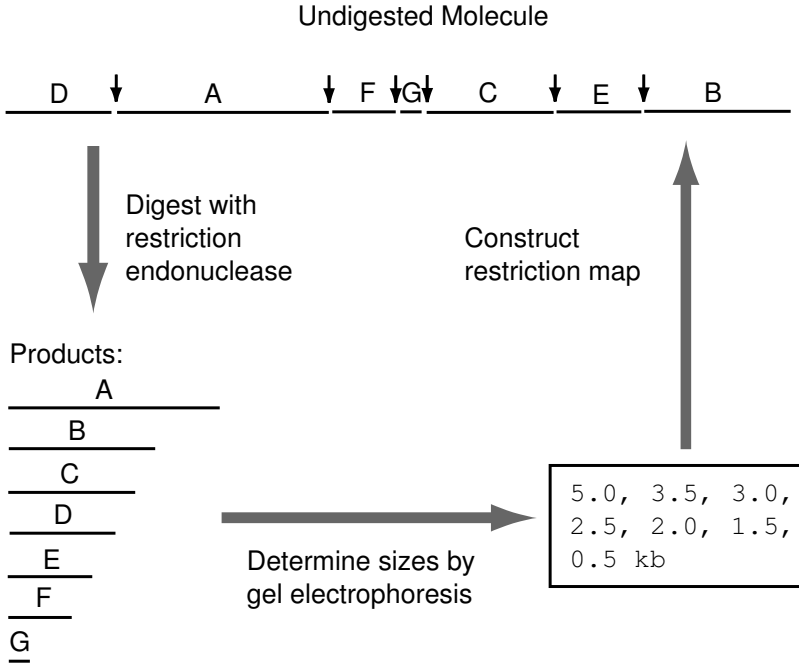


Fig. 3.1. Restriction endonuclease digestion (one enzyme) and the corresponding physical map for a linear molecule. Fragments are labeled in order of decreasing physical size, as would be observed after gel electrophoresis. Restriction sites are indicated by vertical arrows. The order of fragments (D, A, F, G, C, E, B) is originally unknown. A variety of techniques may be employed to determine this order.

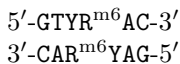
3.1.1 Restriction Endonucleases

Bacteria contain genes for a remarkable collection of enzymes, restriction endonucleases. Here is the setting for their discovery. Bacteriophage lambda grows well on the *E. coli* strain K-12. However, only a very small percentage of bacteriophages grown on strain K-12 can grow well on *E. coli* strain B. Most of the phage DNA that infects strain B is inactivated by fragmentation. However, those few bacteriophages that infect *E. coli* B successfully produce offspring that can infect *E. coli* B (but not K-12) with high efficiency.

The reason for these observations is that the host DNA in *E. coli* B is modified (by methylation) in a particular manner. The invading DNA is not modified in this particular way, and consequently it is broken down. In 1970, Hamilton Smith found that a restriction enzyme that he was studying (*Hind*II) caused cleavage of double-stranded DNA in a short specific nucleotide sequence. Methylation of a base in the sequence prevented this cleavage. The recognition sequence for *Hind*II is

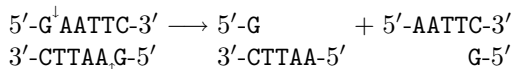


and its methylated form is

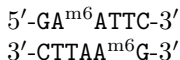


Arrows indicate the position of cleavage at this site. *Hind*II is an example of a Type II restriction endonuclease (Types I and III are also known). Type II enzymes cleave DNA within the recognition sequence. In the case of *Hind*II, the third position (top strand) can be either pyrimidine (Y = C or T) and the fourth position can be either purine (R = A or G). Because the cleavages on the two strands are directly opposite, blunt ends are generated by *Hind*II cleavage, and this enzyme is not used as much in cloning as are other enzymes.

Type II restriction endonucleases that recognize four to eight specific bases are known. Sometimes the specific sequences in the recognition sequence are separated by one or more bases whose identities are indeterminate. For example, *Sfi*I recognizes and cuts within the sequence 5'-GGCCNNNNNGGCC-3' (N can be A, C, G, or T). For the sizes of DNA fragments typical in laboratory cloning experiments, enzymes that recognize six base-pair sequences have been commonly used. For example, *Eco*RI recognizes and cleaves (within the context of DNA on both sides of the sequence—not shown below)



It will not cleave



The particular usefulness of the Type II enzymes was the production of extensions (in this case, the 5' extension AATT), which aided efficient cloning. With synthetic adapter and linker technologies now available, this is less of a consideration. More than 600 restriction endonucleases of different types (I, II, or III) have been described, and over 200 are commercially available.

3.1.2 The Problem in Computational Terms

As described in Chapter 2, the genome of an organism can be represented as a string L_1, \dots, L_n drawn from the alphabet $\mathcal{X} = \{a_1, a_2, a_3, a_4\} = \{\text{A, C, G, T}\}$, where n is the number of base pairs. Given such a string, there are a number of questions that might arise.

- If we were to digest the DNA with a restriction endonuclease such as *Eco*RI, approximately how many fragments would be obtained, and what would be their size distribution?

- Suppose that we observed 761 occurrences of the sequence 5'-GCTGGTGG-3' in a genome that is 50% G+C and 4.6 Mb in size. How does this number compare with the expected number (see Section 2.8)? How would one find the expected number? Expected according to what model?

This chapter provides some tools for answering these sorts of questions. We model the underlying sequence as a string of independent and identically distributed letters (Section 2.3.2) and use this model to find the probability distribution of the number of restriction endonuclease cleavage sites and the distribution of fragment sizes of a restriction digest. We then inquire about the expected frequencies of runs of letters (such as AAAAAA \cdots A tracts). These can be important because of their occurrence in promoter regions. In Chapter 4, we describe the reverse of the digestion problem: Given a set of fragments without knowing the DNA sequence, how do we assemble the **physical map**?

3.2 Modeling the Number of Restriction Sites in DNA

3.2.1 Specifying the Model for a DNA Sequence

If we are given a DNA sample, we usually know something about it—at least which organism it came from and how it was prepared. This means that usually we know its base composition (%G+C) and its approximate molecular weight (especially for bacteriophage or plasmid DNA). In this section, we inquire about the number and distribution of restriction endonuclease recognition sites and about the distribution of resulting restriction fragment lengths after digestion by a specified restriction endonuclease. To describe this mathematically, we need a model. Since our information about the DNA is limited, we select the simplest possible model for our DNA sequence: iid letters.

3.2.2 The Number of Restriction Sites

Now that we know the sequence model, we can proceed with analysis of our **restriction site** problem (e.g., the number and distribution of sites). We have modeled the DNA sequence as a string of iid letters at positions $1, 2, \dots, n$. Restriction endonuclease recognition sequences have length t (4, 5, 6 or 8 typically), and t is very much smaller than n . To begin, our model is going to assume that cleavage can occur between any two successive positions on the DNA. This is wrong in detail because, depending upon where cleavage occurs within the n bases of the recognition sequence (which may differ from enzyme to enzyme), there are positions near the ends of the DNA that are excluded from cleavage. (Also, because the ends of DNA molecules may “fray” or “breathe” for thermodynamic reasons, sites at the ends of molecules may not be cleaved.) However, since t is much smaller than n , the ends of the molecule do not affect the result too much, and our approximation that cleavage can occur between any two positions produces a result that is nearly correct.

We again use X_i to represent the outcome of a trial occurring at position i , but this time X_i does not represent the identity of a base (one of four possible outcomes) but rather whether position i is or is not the beginning of a restriction site. That is,

$$X_i = \begin{cases} 1, & \text{if base } i \text{ is the start of a restriction site,} \\ 0, & \text{if not.} \end{cases} \quad (3.1)$$

We denote by p the probability that any position i is the beginning of a restriction site. The outcomes are then

$$X_i = \begin{cases} 1, & \text{with probability } p, \\ 0, & \text{with probability } 1 - p. \end{cases}$$

If this were a coin-tossing experiment, “heads” might be coded as 1, “tails” might be coded as 0, and X_i would represent the outcome of the i th toss. If the coin were fair, p would be equal to 0.5. For the case of restriction sites on DNA, p depends upon the base composition of the DNA and upon the identity of the restriction endonuclease. For example, suppose that the restriction endonuclease is *EcoRI*, with recognition sequence 5'-GAATTC-3'. (The site really recognized is duplex DNA, with the sequence of the other strand determined by the Watson-Crick base-pairing rules.) Suppose further that the DNA has equal proportions of A, C, G, and T. The probability that any position is the beginning of a site is the probability that this first position is G, the next one is A, the next one is A, the next one is T, the next one is T, and the last one is C. Since, by the iid model, the identity of a letter at any position is independent of the identity of letters at any other position, we see from the multiplication rule (2.6) that

$$p = \mathbb{P}(\text{GAATTC}) = \mathbb{P}(\text{G})\mathbb{P}(\text{A})\mathbb{P}(\text{A})\mathbb{P}(\text{T})\mathbb{P}(\text{T})\mathbb{P}(\text{C}) = (0.25)^6 \sim 0.00024.$$

Notice that p is small, a fact that becomes important later.

The appearance of restriction sites along the molecule is represented by the string X_1, X_2, \dots, X_n , and the number of restriction sites is $N = X_1 + X_2 + \dots + X_m$, where $m = n - 5$. The sum has m terms in it because a restriction site of length 6 cannot begin in the last five positions of the sequence, as there aren't enough bases to fit it in. As we noted earlier, such end effects are not significant when the sequence is long, so for simplicity of exposition we take $m = n$ in what follows. What really interests us is the number of “successes” (restriction sites) in n trials. If X_1, X_2, \dots, X_n were independent of one another, then the probability distribution of N would be known from the result in Section 2.3.4. N would have a binomial distribution (2.17) with parameters n and p ; the expected number of sites would therefore be np and the variance $np(1 - p)$. We remark that the X_i are not in fact independent of one another (because of overlaps in the patterns corresponding to X_i and X_{i+1} , for example). The binomial approximation often works well nonetheless.

Probabilities of events can be computed using the probability distribution in (2.17), but as we saw in Section 2.3.3, this can be cumbersome. In the following sections, we describe two other approximations that can be used for computing probabilities for the number of successes in a binomial experiment. Before doing this, we assess how well our very simplified model behaves by comparing its predictions with data from bacteriophage lambda.

3.2.3 Test with Data

To test the adequacy of the iid model for the DNA sequence and the distribution of restriction sites, we compare the number of sites predicted under the model with the observed number of restriction sites (based upon the experimentally measured DNA sequence). For this comparison, we use restriction endonucleases that recognize four base-pair sequences because this allows easy comparison for most or all of the 16 possible 4 bp **palindromes**. (Remember that most restriction endonucleases having an even-numbered recognition sequence display **inverted repetition** in that sequence, so there are only two independently variable sites for a 4 bp palindrome; therefore the number of different 4 bp palindromes is $4^2 = 16$.) In the example in Table 3.1, the lambda DNA sequence is 48,502 bp long, and we use a model with the observed duplex DNA frequencies $p_A = p_T = 0.2507$, $p_C = p_G = 0.2493$.

Table 3.1. Comparison of observed and expected numbers of restriction enzyme cleavage sites for bacteriophage lambda DNA. (The data were taken from GenBank nucleotide sequence file NC_001416 and from the New England Biolabs online catalog (<http://www.neb.com-Restriction Maps/Frequencies of Restriction Sites>).

Enzyme	Recognition sequence	p	$\mathbb{E}N$	$\text{Var}N$	Observed number
<i>AluI</i>	AGCT	0.00391	190	189	143
<i>BfaI</i>	CTAG	0.00391	190	189	13*
<i>BstUI</i>	CGCG	0.00386	187	186	157
<i>HaeIII</i>	GGCC	0.00386	187	186	149
<i>HpaII</i>	CCGG	0.00386	187	186	328*
<i>MboI</i>	GATC	0.00391	190	189	116*
<i>MseI</i>	TTAA	0.00395	192	191	195
<i>NlaIII</i>	CATG	0.00391	190	189	181
<i>RsaI</i>	GTAC	0.00391	190	189	113*
<i>TaqI</i>	TCGA	0.00391	190	189	121*

Recall that the standard deviation is the square root of the variance, which in all cases above is around 14. In most cases, the observed number of sites is approximately as predicted, suggesting that the iid model adequately describes the number of restriction sites for lambda DNA. There are five enzymes

(indicated by *) whose site numbers in lambda are well over three standard deviations away from the predicted value. If this were consistently observed for other bacteriophages of *E. coli* and for the *E. coli* chromosome, then we might hypothesize that the deficiency of these recognition sequences may reflect some biochemical feature of the organism (e.g., peculiarities of the DNA repair system).

3.2.4 Poisson Approximation to the Binomial Distribution

In preparation for looking at the distribution of restriction fragment lengths, we introduce an approximate formula for $\mathbb{P}(N = j)$ when N has a binomial distribution with parameters n and p . Recall that for the restriction site example, p depends upon the particular restriction endonuclease and upon the base composition of the DNA. For example, we showed above for *EcoRI* that $p = 0.00024$ for DNA that equal frequencies of the four bases. For a molecule that is 50,000 bp long, there would be $50,000 \times 0.00024 = 12$ sites expected according to our model. Notice that because p is very small, the number of sites is small compared with the length of the molecule. This means that $\text{Var}N = np(1 - p)$ will be very nearly equal to $\mathbb{E}N = np$. Contrast this with a fair coin-tossing experiment, where $p = 0.5$. In that case, if we were to perform, say, 300 coin tosses, we would have $\mathbb{E}N = 300 \times 0.5 = 150$, which is much larger than $\text{Var}N = 300 \times 0.5 \times (1 - 0.5) = 75$. In what follows, we assume that n is large and p is small, and we set $\lambda = np$.

In (2.17), we saw that for $j = 0, 1, \dots, n$,

$$\mathbb{P}(N = j) = \frac{n!}{(n-j)!j!} p^j (1-p)^{n-j}.$$

Now we massage the equation algebraically, factoring the various terms and canceling in a manner that will seem reasonable a bit later. First, write

$$\mathbb{P}(N = j) = \frac{n(n-1)(n-2)\cdots(n-j+1)}{j!(1-p)^j} p^j (1-p)^n.$$

Note that there are j terms involving n in the numerator. In the cases in which we are interested, the expected number of sites is small compared with the length of the molecule (so values of j that are relevant are small compared with the length of the molecule, n). This means that

$$n(n-1)(n-2)\cdots(n-j+1) \approx n^j \text{ and } (1-p)^j \approx 1.$$

Substituting these approximations into the equation for $\mathbb{P}(N = j)$ and using $\lambda = np$, we get

$$\mathbb{P}(N = j) \approx \frac{(np)^j}{j!} (1-p)^n = \frac{\lambda^j}{j!} \left(1 - \frac{\lambda}{n}\right)^n.$$

Now recall a result from calculus that, for any x ,

$$\lim_{n \rightarrow \infty} \left(1 - \frac{x}{n}\right)^n = e^{-x}.$$

Since n is large (often more than 10^4), we replace $(1 - \lambda/n)^n$ by $e^{-\lambda}$ to get our final approximation in the form

$$\mathbb{P}(N = j) \approx \frac{\lambda^j}{j!} e^{-\lambda}, \quad j = 0, 1, 2, \dots$$

Some of you will recognize that this is the formula for the Poisson probability distribution with parameter $\lambda = np$. We say that a random variable N taking values in $\{0, 1, 2, \dots\}$ has a Poisson distribution with parameter λ if

$$\mathbb{P}(N = j) = \frac{\lambda^j}{j!} e^{-\lambda}, \quad j = 0, 1, 2, \dots \quad (3.2)$$

It can be shown that $\mathbb{E}N = \text{Var}N = \lambda$ for the Poisson distribution (see Exercise 1).

Computational Example 3.1: Poisson approximation for the binomial

We have just illustrated the well-known result that the binomial distribution can be approximated by the Poisson distribution if np is small or of moderate size and p is small. To show how this approximation may be used, we estimate the probability that there are no more than two *EcoRI* sites in a DNA molecule of length 10,000, assuming equal base frequencies.

Earlier we calculated that $p = 0.00024$ in this case. Therefore $\lambda = np$ is calculated to be 2.4. The problem is to calculate $\mathbb{P}(N \leq 2)$. Using the Poisson distribution in (3.2), we get

$$\mathbb{P}(N \leq 2) \approx \frac{\lambda^0}{0!} e^{-\lambda} + \frac{\lambda^1}{1!} e^{-\lambda} + \frac{\lambda^2}{2!} e^{-\lambda} \approx 0.570.$$

This can also be evaluated using the R command `ppois`, which gives the distribution function of the Poisson random variable.

```
> ppois(2,2.4)
[1] 0.5697087
```

In other words, more than half the time, molecules of length 10,000 and uniform base frequencies will be cut by *EcoRI* two times or less.

3.2.5 The Poisson Process

There is a more general version of the Poisson distribution that is very useful. It generalizes n into “length” and p into “rate.” The mean of the corresponding

Poisson distribution is length \times rate. We suppose that events (which were restriction sites above) are occurring on a line at rate μ ; then

$$\mathbb{P}(k \text{ events in } (x, x + l)) = \frac{e^{-\mu l} (\mu l)^k}{k!}, \quad k = 0, 1, 2, \dots$$

If there is more than one interval, the lengths of the intervals simply add,

$$\mathbb{P}(k \text{ events in } (x, x + l_1) \cup (y, y + l_2)) = \frac{e^{-\mu(l_1 + l_2)} (\mu(l_1 + l_2))^k}{k!}, \quad k = 0, 1, 2, \dots,$$

as long as the intervals are disjoint (i.e., $x < x + l_1 \leq y < y + l_2$).

Poisson processes have events occurring “uniformly” along the line. It is easy to generalize the idea to area or volume. For example, lightning strikes might occur in a county according to a Poisson process. Then μ might be in units of strikes per square foot (i.e., events/area), and l would be in units of square feet (area).

3.3 Continuous Random Variables

In the previous section, we saw that the probability of finding $N = j$ restriction sites in a molecule could be represented by the binomial distribution or the Poisson distribution. These are discrete distributions since N takes on integral values. To calculate the probability that N takes on a range of values, for example $\mathbb{P}(N \leq k)$, we calculate a sum,

$$\mathbb{P}(N \leq k) = \sum_{j=0}^k \mathbb{P}(N = j).$$

When a random variable X can take on any value in an interval, we call X *continuous*. The probabilistic behavior of X is then determined by its **probability density function** $f(x)$, $-\infty < x < \infty$. The function f satisfies

$$f(x) \geq 0 \text{ for all } x, \quad \text{and} \quad \int_{-\infty}^{\infty} f(x) dx = 1. \quad (3.3)$$

To compute the probability that X takes values in a set A , we use

$$\mathbb{P}(X \in A) = \int_A f(x) dx;$$

in particular, when A is an interval $(a, b]$, we have

$$\mathbb{P}(a < X \leq b) = \int_a^b f(x) dx.$$

The mean $\mathbb{E}X$ of a continuous random variable X with density f is defined as

$$\mathbb{E}X = \int_{-\infty}^{\infty} xf(x)dx, \quad (3.4)$$

and the variance of X is calculated as

$$\text{Var}(X) = \int_{-\infty}^{\infty} (x - \mu)^2 f(x)dx = \int_{-\infty}^{\infty} x^2 f(x)dx - \mu^2, \quad (3.5)$$

where $\mu = \mathbb{E}X$. These definitions are the analogs of those in (2.7) and (2.11), respectively.

In Section 2.4, we met our first continuous random variable, the one distributed uniformly over $(0, 1)$. A random variable U is said to have the **uniform distribution** on (a, b) if its density function is

$$f(x) = \begin{cases} \frac{1}{b-a}, & a < x \leq b; \\ 0, & \text{otherwise.} \end{cases} \quad (3.6)$$

It is easy to verify that $\mathbb{E}U = (a + b)/2$ and $\text{Var} U = (b - a)^2/12$; see Exercise (2).

In the following chapters, we meet several other continuous random variables. It is convenient at this point to introduce two important examples, the exponential and the normal. The *exponential random variable with parameter λ* has probability density function

$$f(x) = \begin{cases} \lambda e^{-\lambda x}, & x \geq 0; \\ 0, & \text{otherwise.} \end{cases} \quad (3.7)$$

If X has the exponential distribution with parameter λ , then

$$\mathbb{P}(a < X \leq b) = \int_a^b \lambda e^{-\lambda x} dx = e^{-\lambda a} - e^{-\lambda b}.$$

The mean of X is

$$\mathbb{E}X = \int_0^{\infty} x \lambda e^{-\lambda x} dx = 1/\lambda,$$

while

$$\mathbb{E}X^2 = \int_0^{\infty} x^2 \lambda e^{-\lambda x} dx = 2/\lambda^2.$$

It follows that $\text{Var}X = 1/\lambda^2$.

The random variable Z is said to have a *standard normal distribution* if its probability density function is given by

$$\phi(z) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}z^2}, \quad -\infty < z < \infty. \quad (3.8)$$

The probability $\Phi(z)$ that $Z \leq z$ is given by

$$\Phi(z) = \mathbb{P}(Z \leq z) = \int_{-\infty}^z \phi(x) dx, \quad (3.9)$$

and the probability that Z lies in the interval (a, b) is

$$\mathbb{P}(a < Z \leq b) = \int_a^b \phi(z) dz = \Phi(b) - \Phi(a), \quad a < b.$$

The function $\Phi(z)$ cannot be written in closed form, but its values can be computed numerically. For our purposes, we can use R to do this computation. As shown in the box below, we find for example that

$$\mathbb{P}(-1.96 < Z < 1.96) = 0.95,$$

and

$$\mathbb{P}(Z > -1) = 1 - \Phi(-1) \approx 0.841.$$

If Z has the standard normal distribution, we write $Z \sim N(0, 1)$. It can be shown that if $Z \sim N(0, 1)$, then $\mathbb{E}Z = 0$, $\text{Var}Z = 1$.

If $Z \sim N(0, 1)$, the random variable $X = \mu + \sigma Z$ is said to have a Normal distribution with mean μ and variance σ^2 ; we write $X \sim N(\mu, \sigma^2)$ in this case. To calculate probabilities for such X , we use the fact that if $X \sim N(\mu, \sigma^2)$, then $Z = (X - \mu)/\sigma \sim N(0, 1)$. That is,

$$\mathbb{P}(X \leq x) = \mathbb{P}\left(\frac{X - \mu}{\sigma} \leq \frac{x - \mu}{\sigma}\right) = \Phi\left(\frac{x - \mu}{\sigma}\right).$$

For example, if $X \sim N(1, 4)$, then

$$\mathbb{P}(X \leq 1.96) = \Phi\left(\frac{1.96 - 1}{2}\right) = \Phi(0.48) \approx 0.684$$

and

$$\mathbb{P}(X \leq -1.96) = \Phi\left(\frac{-1.96 - 1}{2}\right) = \Phi(-1.48) \approx 0.069,$$

so that

$$\mathbb{P}(-1.96 < X \leq 1.96) = 0.684 - 0.069 = 0.615.$$

These calculations can be performed simply in R, as shown in the box below.

Computational Example 3.2: Using R to compute probabilities for the normal distribution

To calculate probabilities for the standard normal distribution, we use the R function `pnorm(z)`, which calculates

$$\text{pnorm}(z) = \Phi(z) = \int_{-\infty}^z \phi(z) dz.$$

```

> pnorm(1.96,0,1) - pnorm(-1.96,0,1) # P(-1.96 < Z < 1.96)
[1] 0.9500042
> 1 - pnorm(-1) # P(Z > -1)
[1] 0.8413447

```

To calculate probabilities for the normal random variable X with mean $\mu = \mathbf{m}$ and standard deviation $\sigma = \mathbf{s}$, we use the function `pnorm(x,m,s)`, where $\mathbb{P}(X \leq x) = \text{pnorm}(x, \mathbf{m}, \mathbf{s})$. For example,

```

> pnorm(1.96,1,2)
[1] 0.6843863
> pnorm(-1.96,1,2)
[1] 0.06943662
> pnorm(1.96,1,2) - pnorm(-1.96,1,2)
[1] 0.6149497

```

as shown in the text.

The concept of independence of continuous random variables X_1, \dots, X_n is just the same as in the discrete case; the formal definition is given in (2.5).

3.4 The Central Limit Theorem

In this section we give another useful approximation that allows us to calculate probabilities. This result, known as the **Central Limit Theorem**, applies to sums or averages of independent, identically distributed random variables. Assume that X_1, X_2, \dots, X_n are iid with mean μ and variance σ^2 , and denote their sample average by \bar{X}_n , where

$$\bar{X}_n = \frac{1}{n} (X_1 + \dots + X_n).$$

From (2.9) and (2.14), we know that

$$\mathbb{E}\bar{X}_n = \mu, \text{Var}\bar{X}_n = \frac{\sigma^2}{n},$$

and therefore that

$$\mathbb{E}\left(\frac{\bar{X}_n - \mu}{\sigma/\sqrt{n}}\right) = 0, \text{Var}\left(\frac{\bar{X}_n - \mu}{\sigma/\sqrt{n}}\right) = 1.$$

The Central Limit Theorem states that if the sample size n is large,

$$\mathbb{P}\left(a \leq \frac{\bar{X}_n - \mu}{\sigma/\sqrt{n}} \leq b\right) \approx \Phi(b) - \Phi(a), \quad (3.10)$$

the approximation improving as the sample size increases. The terms on the right involve the standard normal distribution function in (3.9). We note that

$$\frac{\bar{X}_n - \mu}{\sigma/\sqrt{n}} = \frac{\sum_{i=1}^n X_i - n\mu}{\sigma\sqrt{n}},$$

obtained by multiplying the top and bottom of the left-hand side by n . This gives a version of the Central Limit Theorem for sums:

$$\mathbb{P}\left(a \leq \frac{\sum_{i=1}^n X_i - n\mu}{\sigma\sqrt{n}} \leq b\right) \approx \Phi(b) - \Phi(a). \quad (3.11)$$

To show the Central Limit Theorem in action, we use simulated binomial data (see Figure 2.1). In the box below, we show how to use R to superimpose the standard normal density (3.8) on the histogram of the simulated values. In each case, we **standardize** the simulated binomial random variables by subtracting the mean and then dividing by the standard deviation.

Computational Example 3.3: Illustrating the Central Limit Theorem

First, we generate 1000 binomial observations with $n = 25$, $p = 0.25$, and then standardize them.

```
> bin25 <- rbinom(1000,25,0.25)
> 25 * 0.25      # Calculate the mean
[1] 6.25
> sqrt(25 * 0.25 * 0.75) # Calculate standard deviation
[1] 2.165064
> bin25 <- (bin25 - 6.25)/2.1651 # Standardize observations
```

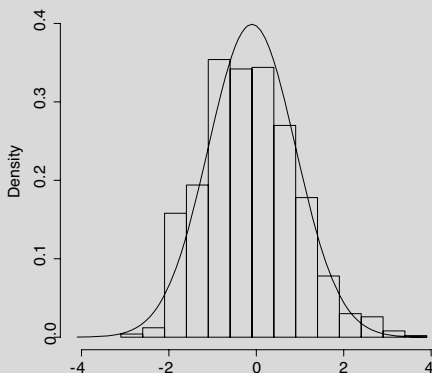


Fig. 3.2. Histogram of 1000 standardized replicates of a binomial random variable with $n = 25$ trials and success probability $p = 0.25$. The standard normal density is superimposed.

In Fig. 3.2 we plot a histogram of the observations and then superimpose the standard normal density. The code to achieve this is given below.

```
> hist(bin25,xlim=c(-4,4),ylim=c(0.0,0.4),prob=TRUE,
+      xlab="Sample size 25",main="")
> x<-seq(-4,4,0.1) # Generate grid of points
                   # from -4 to 4 in steps of 0.1
> lines(x,dnorm(x))
```

Now we return to the question posed at the end of Section 2.4: What is the probability that a random, uniformly distributed DNA sequence of length 1000 contains at least 280 A s? That is, we want to estimate the quantity $\mathbb{P}(N \geq 280)$ when N has a binomial distribution with parameters $n = 1000$ and $p = 0.25$. We saw in (2.3) that N is the sum of iid random variables, so we can use (3.11) to estimate $\mathbb{P}(N \geq 280)$.

To do this, we calculate the mean and standard deviation for N in (2.3) having $n = 1000$, obtaining

$$\mathbb{E}(N) = n\mu = 1000 \times 0.25 = 250,$$

and

$$\text{sd}(N) = \sqrt{n}\sigma = \sqrt{1000 \times \frac{1}{4} \times \frac{3}{4}} \approx 13.693.$$

Hence

$$\mathbb{P}(N \geq 280) = \mathbb{P}\left(\frac{N - 250}{13.693} > \frac{280 - 250}{13.693}\right) \approx \mathbb{P}(Z > 2.19) = 0.014.$$

This is in good agreement with the theoretical result 0.016 cited earlier.

3.4.1 Confidence Interval for Binomial Proportion

In Section 2.4, we used simulation to estimate a proportion. The parameter of interest was $p = \mathbb{P}(N \geq 280)$. In $n = 10,000$ repeated simulations of 1000 bp under the iid model with equal base frequencies, we observed 149 occasions where the number N of As in the sequence exceeded 280. Each of these we can call a “success” and the remaining trials “failures.” How accurately have we estimated the success probability p ?

To answer this question, we can use the Central Limit Theorem again. Write \hat{p} for the observed proportion of successes in the 10,000 trials. Using our results about the binomial distribution, we know that the expected value $\mathbb{E}\hat{p} = p$, and $\text{Var}\hat{p} = p(1 - p)/10,000$. Given the large number of trials, we know that

$$\mathbb{P}\left(-1.96 < \frac{\hat{p} - p}{\sqrt{p(1 - p)/10,000}} < 1.96\right) \approx 0.95.$$

Given that the sample size is very large, we expect $\hat{p} \approx p$, so that

$$\mathbb{P}\left(-1.96 < \frac{\hat{p} - p}{\sqrt{\hat{p}(1 - \hat{p})/10,000}} < 1.96\right) \approx 0.95$$

as well. This statement can be rewritten in the form

$$\mathbb{P}\left(\hat{p} - 1.96\sqrt{\frac{\hat{p}(1 - \hat{p})}{10,000}} < p < \hat{p} + 1.96\sqrt{\frac{\hat{p}(1 - \hat{p})}{10,000}}\right) \approx 0.95.$$

This says that if we repeat our whole simulation many times, the random interval

$$\left(\hat{p} - 1.96\sqrt{\frac{\hat{p}(1 - \hat{p})}{10,000}}, \hat{p} + 1.96\sqrt{\frac{\hat{p}(1 - \hat{p})}{10,000}}\right) \quad (3.12)$$

will cover the true value p about 95% of the time. We call this a *95% confidence interval* for p . In our example, we got $\hat{p} = 0.0149$ and a 95% confidence interval of $(0.0125, 0.0173)$, which does indeed cover the true value of $p = 0.0164$.

To obtain confidence intervals with confidence levels other than 0.95, all that has to be done is to replace the value 1.96 in (3.12) with the appropriate value found from the standard normal distribution. For example, for a 90% confidence interval use 1.645.

Computational Example 3.4: p-values It is common practice in much of statistics to measure one's surprise about the outcome of a series of experiments by reporting the **p-value** of the result. The p-value is the probability of obtaining a result more extreme than the value observed in the experiment. For example, suppose we tossed a fair coin 100 times and observed that the number of heads N (which has a binomial distribution with parameters $n = 100$ and $p = 0.5$) was 65. Then the p-value would be

$$\mathbb{P}(N \geq 65) \approx \mathbb{P}(Z \geq 3) \approx 0.0013,$$

where Z has a standard normal distribution. This result follows from the Central Limit Theorem and the fact that N has mean 50 and standard deviation 5.

Sometimes p-values are two-sided. In the coin-tossing example, we would calculate the probability that values more than three standard deviations from the mean are observed. The p-value in this case would be $2 \times 0.0013 \approx 0.003$.

3.4.2 Maximum Likelihood Estimation

In Section 2.6.1, we gave a simple method for estimating the transition matrix of a Markov chain from a sequence of observations of that chain. This

is an example of *parameter estimation*, which arises frequently in statistical applications. The setting usually involves trying to find “good” values for the parameters of a probability model using a set of observations. We illustrate one method of parameter estimation in the context of estimating a binomial proportion.

In this case, the (unknown) parameter of interest is the success probability, p . We estimated this in the previous section by using the observed proportion of successes in the n trials—obviously a sensible thing to do. Here we provide another rationale for this choice based on *the method of maximum likelihood*. Suppose then that we observed k successes in n tosses of a coin with success probability p . For a given value of p , the chance of this observation is

$$L(p) = \binom{n}{k} p^k (1-p)^{n-k}.$$

Note that we are treating k as fixed here. (It was the value we observed in the experiment.) The maximum likelihood approach estimates the parameter p by using that value of p that maximizes the **likelihood function** $L(p)$.

Elementary considerations show that the value of p that maximizes $L(p)$ also maximizes the log-likelihood function $l(p) = \log L(p)$; the latter is often a simpler function to optimize. To maximize

$$l(p) = \log \binom{n}{k} + k \log p + (n-k) \log(1-p),$$

we solve the equation

$$\frac{dl(p)}{dp} = 0,$$

obtaining the equation

$$\frac{k}{p} - \frac{n-k}{1-p} = 0.$$

The solution of this equation is obtained as $p = k/n$, the observed fraction of successes in the n trials. The value $\hat{p} = k/n$ is called the **maximum likelihood estimator (MLE)** of the parameter p . In this case, the maximum likelihood method has given the same estimator as we derived intuitively earlier. (We note that we should check that \hat{p} does indeed give a *maximum* of the likelihood function. You should check that if you compute the value of $\frac{d^2l(p)}{dp^2}$ at the point $p = \hat{p}$, you get a negative value, as required.)

The maximum likelihood approach for parameter estimation is one of the basic methods in statistics. The general scheme is to write down the likelihood function, the probability (or the probability mass function) of the observations, viewed as a function of the unknown parameters in the model and then maximize over the parameters. This can sometimes be done by simple calculus but more often involves numerical approaches. This aspect is not discussed further in this book, but we note that many statistics packages have flexible tools for such numerical analysis; R is no exception.

3.5 Restriction Fragment Length Distributions

We assume that restriction sites occur according to a Poisson process with rate λ per bp. Then the probability of k sites in an interval of length l bp is

$$\mathbb{P}(N = k) = \frac{e^{-\lambda l}(\lambda l)^k}{k!}, \quad k = 0, 1, 2, \dots \quad (3.13)$$

We can also calculate the probability that a restriction fragment length X is larger than x . If there is a site at y , then the length of that fragment is greater than x if there are no events in the interval $(y, y + x)$. From (3.13), this has probability

$$\mathbb{P}(X > x) = \mathbb{P}(\text{no events in } (y, y + x)) = e^{-\lambda x}, \quad x > 0.$$

It follows that

$$\mathbb{P}(X \leq x) = \int_0^x f(y)dy = 1 - e^{-\lambda x},$$

and the density function for X is then

$$f(x) = \lambda e^{-\lambda x}, \quad x > 0.$$

Hence, recalling (3.7), the distance between restriction sites has an exponential distribution with parameter λ ; the mean length is $1/\lambda$.

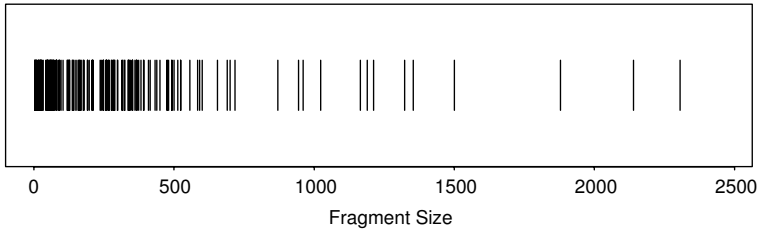
3.5.1 Application to Data

To develop intuition about what this means, let's look at a specific example: the result of digesting bacteriophage lambda with *AluI*. We already used this example for comparison with our model for estimation of the expected number of fragments (Table 3.1). The restriction fragment lengths and a histogram of these lengths are shown in Fig. 3.3. The data come from the New England Biolabs catalog; check the links at <http://www.neb.com>.

We see that there are more fragments of shorter lengths and that the number having longer lengths decreases, as predicted by the model. We can also estimate the proportion of fragments we would expect to see with lengths greater than $d = 1000$ bp (say). For the model used in Table 3.1, we have $n = 48,502$, $p = 0.003906$, so that $x = d/n = 0.0206$ and $\lambda = np = 189.45$. Thus the probability of a fragment being longer than 1000 bp is $e^{-\lambda x} = e^{-3.903} = 0.020$.

Notice from Fig. 3.3 that in the lambda data there were ten fragments with lengths greater than 1000 bp, whereas we would have predicted $143 \times 0.020 \approx 2.9$. There is some evidence that our simple probability model does not describe the longer fragments very well. In the next section, we describe a simulation approach that can be used to study fragment lengths generated by far more complicated probability models.

A.



B.

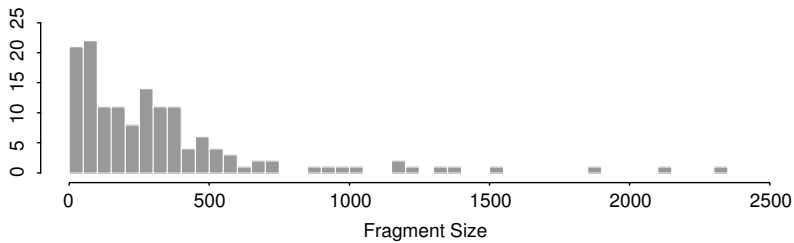


Fig. 3.3. Fragments produced by *AluI* digestion of bacteriophage lambda DNA. Panel A: Lengths of individual fragments. Panel B: Histogram of fragment sizes.

3.5.2 Simulating Restriction Fragment Lengths

In the preceding section, we showed that the restriction fragment length distribution should be approximately exponential, and we demonstrated that a particular real example did indeed resemble this distribution. But what would we actually see for a particular sequence conforming to the iid model? In other words, if we *simulated* a sequence using the iid model, we could compute the fragment sizes in this simulated sequence and visualize the result in a manner similar to what is seen in the actual case in Fig. 3.3. The details of this R exercise are given in Computational Example 3.5.

Computational Example 3.5: Simulating restriction site distributions

We assume the iid model for the sequence, with uniform base probabilities $p_A = p_C = p_G = p_T = 0.25$. Comparison with the data for lambda DNA in Table 3.1 shows that this approximation is not too bad. We generate a sequence having 48,500 positions (close to the length of lambda DNA). As earlier, we code the bases as follows: A=1, C=2, G=3, and T=4. The following is the result of an R session simulating the sequence:

```
> x<-c(1:4)
```

```

> propn<-c(0.25,0.25,0.25,0.25)
> seq2<-sample(x,48500,replace=T,prob=propn)
> seq2[1:15]
[1] 2 2 4 2 1 4 3 4 3 1 3 2 1 1 4
> length(seq2[])
[1] 48500

```

The first line defines a vector with elements 1, 2, 3, and 4. The second line defines the probabilities of each base, corresponding to our probability model. The third line samples 48,500 times with replacement according to the frequencies in `propn`. The last two commands show that the simulation did run (for the first 15 elements) and the length of our simulated iid sequence string in which we seek restriction sites. Other base frequencies can be simulated by changing the entries in `propn`.

Locating the restriction sites

The following function operating under R will identify the restriction sites in a sequence string, with bases coded numerically:

```

> rsite <- function(inseq, seq){
  # inseq: vector containing input DNA sequence,
  # A=1, C=2, G=3, and T=4
  # seq: vector for the restriction site, length m
  # Make/initialize vector to hold site
  # positions found in inseq
  xxx <- rep(0, length(inseq))
  m <- length(seq)
  #To record whether position of inseq matches seq
  truth<-rep(0, m)
  # Check each position to see if a site starts there.
  for(i in 1:(length(inseq) - (length(seq) - 1))) {
    for(j in 1:m) {
      if(inseq[i + j - 1] == seq[j]) {
        truth[j] <- 1 # Record match to jth position.
      }
    }
  }
  if(sum(truth[]) == m){# Check whether all positions match
    xxx[i] <- i      # Record site if all positions match
  }
  truth <- rep(0, m) # Reinitialize for next loop cycle
}
# Write vector of restriction site positions stored in xxx
L <- xxx[xxx > 0]
return(L)
}

```

The restriction sites we look for are for *AluI*, AGCT. We code this as [1 3 2 4], and then use the `rsite` function to find the sites and write the result into `alu.map`.

```
> alu1 <- c(1,3,2,4)
> alu.map <- rsite(seq2,alu1)
```

The nested loops cause this to take a bit of time to run. The output is the initial positions of all of the *AluI* sites in the simulated sequence. Checking the output (length and first ten terms):

```
> length(alu.map)
[1] 184
> alu.map[1:10]
[1] 645 915 1076 1790 1836 1957 2100 2566 2881 2980
```

Note that the prediction from the mathematical model was 190 sites: We found 184 for this particular string. (To estimate the expected number, we would need to simulate many strings and report the mean number of sites detected.) We judge that the model agrees well with the simulation (so far!).

Obtaining and displaying fragment lengths and fragment length distribution

We obtain the fragment lengths by subtracting positions of successive sites. Since the molecule is linear, we need to deal with the ends. The function `flengthr`, written in R, does these things for us.

```
> flengthr <- function(rmap, N){
  #rmap is a vector of restriction sites for a linear molecule
  # N is the length of the molecule
  frags<-rep(0, length(rmap))
  # Vector for subtraction results: elements initialized to 0
  rmap<-c(rmap,N)
  # Adds length of molecule for calculation of end piece.
  for(i in 1:(length(rmap)-1)){
    frags[i] <- rmap[i+1]-rmap[i]}
  frags <- c(rmap[1],frags) # First term is left end piece
  return(frags)
}
> alu.frag <- flengthr(alu.map,48500)
> alu.frag[1:10]
[1] 645 270 161 714 46 121 143 466 315 99
```

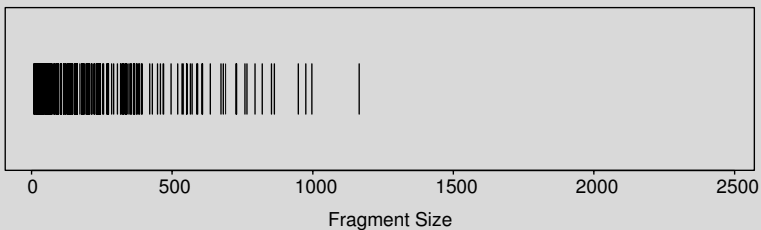
The two lines above run the function and display the first ten fragment lengths. You can verify that the second element in `alu.frag` is the difference between the first two elements in `alu.map`.

```
> max(alu.frag[])
[1] 1475
> min(alu.frag[])
[1] 5
```

The four lines above display the largest and smallest fragments.

```
> length(alu.frag[])
[1] 185
> sum(alu.frag[])
[1] 48500
```

A.



B.

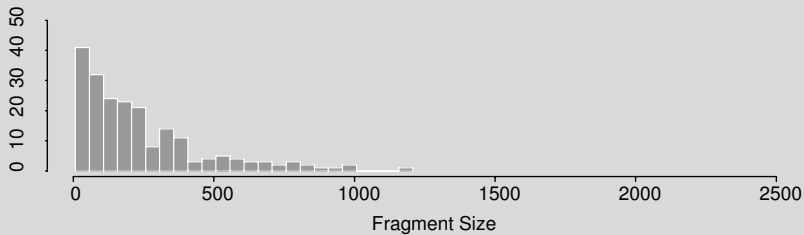


Fig. 3.4. Fragments predicted for an *AluI* digestion of simulated DNA. Panel A: Lengths of individual fragments. Panel B: Histogram of fragment sizes.

The first two lines verify that the number of fragments is what it should be. (n sites in a linear molecule generate $n + 1$ fragments.) The second two lines show that the fragment sizes sum to the length of the molecule. These results are expected if the function does what it should do. The data in `alu.frag` are plotted in Fig. 3.4, as was done in Fig. 3.3.

The particular simulated sequence that we generated yields a distribution of restriction fragment lengths that looks similar to the distribution observed for bacteriophage lambda DNA fragments (Fig. 3.3). However, note the different scales for the ordinates in the lower panels for the two figures and the

greater number of large fragments in the case of bacteriophage lambda DNA. To determine whether the distribution for lambda DNA differs significantly from the mathematical model (exponential distribution), we could break up the length axis into a series of “bins” and calculate the expected number of fragments in each bin by using the exponential density. This would create the entries for a histogram based on the mathematical model. We could then compare the observed distribution of fragments from lambda DNA (using the same bin boundaries) to the expected distribution from the model by using the χ^2 test, for example. Further details are given in Exercise 13.

3.6 k -word Occurrences

The statistical principles learned in this and the previous chapter can be applied to other practical problems, such as discovering functional sites in promoter sequences. Recall from Section 1.3.4 that promoters are gene regions where RNA polymerase binds to initiate transcription. We wish to find k -words that distinguish promoter sequences from average genomic sequences. Because promoters are related by function, we expect to observe k -words that are over-represented within the promoter set compared with a suitable null set. These k -words can help identify DNA “signals” required for promoter function. (DNA signals are described in detail in Chapter 9.) Using the approaches of Chapter 2, we determine expected k -word frequencies and compare them to the observed frequencies. Distributions presented in this chapter are used to test whether over-represented k -words appear with significantly higher frequencies.

Consider N promoter sequences of length L bp, which we denote by S_i for $i = 1, \dots, N$ (Table C.2). The null set might consist of N strings of L iid letters, each letter having the same probability of occurrence as the letter frequencies in genomic DNA as a whole. For the purposes of the discussion here, we take a small word size, $k = 4$, so that there are 256 possible k -words. With no a priori knowledge of conserved patterns, we must examine all 256 words. We ask whether there are an unusual number of occurrences of each word in the promoter regions.

For the 49 promoter sequences shown in Table C.2 in Appendix C, we first evaluate the most abundant observed k -words and their expected values for $k = 4$ using R for the computation described in Computational Example 3.6. The expectation of each word according to the null (iid) model is easy to calculate if words are overlapping. For example, if X_w denotes the number of occurrences of word w in the whole set of sequences, then for $w = \text{ACGT}$,

$$\mathbb{P}(w = \text{ACGT}) = p_A p_C p_G p_T$$

$$\mathbb{E}(\# \text{ times } w \text{ appears in } S_i) = (L - 4 + 1)p_A p_C p_G p_T$$

and the expected number of occurrences in N such sequences is

$$\mathbb{E}(X_w) = N(L - 4 + 1)p_A p_C p_G p_T. \quad (3.14)$$

Computational Example 3.6: Counting k -words in promoter sequences

The data in Table C.2 are stripped of row labels, and A, C, G, and T are coded numerically as 1, 2, 3, and 4 separated by spaces, as has been our usual practice. The result is saved as a text file, `Ec.table.txt`, which is then read into a matrix in R:

```
>ec.prom<-matrix(scan("Ec.table.txt"),nrow=49,byrow=T)
Read 2499 items
```

We must first decide to what we wish to compare the promoters. In this example, we compare them with the average *E. coli* sequence, and in an exercise you will compare them with sequences having the promoter base composition. The base frequencies for the *E. coli* genome are, for A, C, G, and T, respectively, (see Table 2.1):

```
> prob.ec
[1] 0.246 0.254 0.254 0.246
```

These values are employed for calculating the expected value of each k -word and later for simulating sequences under the null model.

Since we are concerned with k -words having $k = 4$, it is convenient to store the results of our calculations in four-dimensional arrays. We can think of these as four different three-dimensional arrays, each labeled 1, 2, 3, 4, but this visualization is not required for handling the array objects under R. We represent the word size, k , as w in this function, so $w = 4$ in our example. Because the base frequencies are, in general, all different, the expected word frequencies are not all identical. The code to generate the expected frequencies is:

```
> expect4.ec<-array(rep(0,4^w), rep(4,w))
> for(i in 1:4){
  for(j in 1:4){
    for(k in 1:4){
      for(m in 1:4){
        expect4[i,j,k,m]<-
+ 48*49*prob.ec[i]*prob.ec[j]*prob.ec[k]*prob.ec[m]
      } } } }
```

The number 49 corresponds to N , the number of sequences, and 48 comes from $L - w + 1 = 51 - 4 + 1$, where 51 is the number of bases listed for each string. The expected frequency of each k -word is read from the corresponding array element. For example, if the word is GATC, the coded word is represented as 3142, and the expected frequency of that word is contained in the `expect4[3,1,4,2]` array element. The function below is used to perform the

word count, accumulating the total counts for each word in a four-dimensional array, `tcount`. The plug-in portion is included for the computation to be performed in the next box. For counting only, we could “comment out” all lines in the plug-in, and delete the `ncount` in the `return()` statement. For now, we just concentrate on `tcount`.

```
Ncount4<-function(seq,w){
  #w=length of word
  tcount<-array(rep(0,4^w), rep(4,w))
  # array[4x4x4x4] to hold word counts, elements set to zero
  ncount<-array(rep(0,4^w), rep(4,w))
  # array[4x4x4x4] holds number of sequences with one or
  # more of each k-word
  N<-length(seq[1,]) #Length of each sequence
  M<-length(seq[,1]) #Number of sequences
  #####
  #Count total number of word occurrences
  for(j in 1:M){ #looping over sequences
    jcount<-array(rep(0,4^w), rep(4,w))
    #array to hold word counts for sequence j
    for(k in 1:(N-w+1)){ #looping over positions
      jcount[seq[j,k],seq[j,k+1],seq[j,k+2],seq[j,k+3]]<-
      jcount[seq[j,k],seq[j,k+1],seq[j,k+2],seq[j,k+3]]+1
      #adds 1 if word at k,k+1,k+2,k+3 appears in sequence j
    }
    tcount<-tcount+jcount
    #Add contribution of j to total
    #####
    #Plug-in: add 1 to ncount if word occurs >= once in j
    for(k in 1:4){
      for(l in 1:4){
        for(m in 1:4){
          for(n in 1:4){
            if(jcount[k,l,m,n]!=0){
              ncount[k,l,m,n]<-ncount[k,l,m,n]+1
            }
          }
        }
      }
    }
    #####
  }
  return(tcount,ncount)
}
```

The word count is performed on the promoter sequences:

```
> prom.count<-Ncount4(ec.prom,4)
> sum(prom.count$count)
[1] 2352
```

The last two lines verify that the expected number of words have been counted: $48 \times 49 = 2352$. Note that since two items are being returned from the computation, we must specify which item in the list we require; in this case, `prom.count$count`. The most frequently occurring word appears 33 times:

```
> max(prom.count$count)
[1] 33
```

We find that ten words appear more than 20 times in this set of promoter sequences:

```
>(1:256)[prom.count$count[prom.count$count[, , ]>20]]
[1] 23 22 21 21 23 24 24 25 22 33
```

By inspection, we identify these words in the output of the array contents and tabulate the result in Table 3.2. Expected values are taken from corresponding elements of `expect4`.

Table 3.2. Observed and expected k -word frequencies in *E. coli* promoter sequences for $k = 4$. Expected values were computed under the iid null model where $p_A = p_T = 0.246$ and $p_G = p_C = 0.254$. Only the ten most abundant words, based on the total number of appearances in all promoter sequences, are shown. Example promoter sequences are shown in Table 9.2.

Word	Observed frequency	Expected frequency
TTTT	33	8.6
CATT	25	8.9
AATT	24	8.6
TAAT	24	8.6
ATTG	23	8.9
TGAA	23	8.9
ATAA	22	8.6
ATTT	22	8.6
TTTA	21	8.6
ATTC	21	8.9

These results suggest that promoters have unusual word composition compared with the iid null model. These words are composed mostly of As and Ts, the most frequent letters in the promoter set. We must determine whether these elevated abundances are statistically significant.

If a k -word is to be identified as significantly over-represented in promoters compared with the null set, we need to know the expected number of occurrences, $\mathbb{E}(X_w)$, and the standard deviation of this number. In other words, we need to know how the values X_w are distributed. A computational approach is to repeatedly simulate sets of N iid sequences, perform the word counts, and then produce a histogram of the observed values of X_w for each word w . The resulting distributions for each word might not be normal, but we could determine thresholds such that an appropriately small fraction of observations (0.001, for example) fall outside this range. If the distributions of the X_w were approximately normal, then a significance level of three standard deviations would correspond to a probability of approximately 0.0013. If there were $N = 100$ sequences, the expected number of words that are three or more standard deviations above the mean would therefore be 0.13.

We do not simply compute $\mathbb{E}(X_w)$ and $\text{Var}(X_w)$ from first principles because both $\mathbb{E}(X_w)$ and $\text{Var}(X_w)$ depend on how the words are counted. For example, if $k = 4$, what is X_w for $w = \text{AAAA}$? If word overlaps are allowed, $X_w = 2$, whereas if word overlaps are not allowed, $X_w = 1$. If $p_A = p_C = p_G = p_T = 0.25$ and word overlaps are allowed, $\mathbb{E}(X_w)$ is identical for each word of length k (see (3.14), but if overlaps are not allowed, $\mathbb{E}(X_w)$ in general differs for different words having the same k . With either way of counting words, $\text{Var}(X_w)$ is not the same for each word. There are basically two approaches for word counting. One is to count all occurrences of the word in the whole set of N regions S_i as we did above, and the second is to count the number of promoter sequences in which the word occurs at least once.

The simple, naive basis we use for deciding whether w occurs with unusual frequency is to take each word w and tabulate the number of promoter sequences N_w in which the word occurs at least once. This alternative statistic conforms to the normal approximation of the binomial distribution. First, we simulate 5000 sequences with letter probabilities corresponding to the *E. coli* genome. We use the simulations to estimate

$$p_w = \mathbb{P}(w \text{ occurs at least once in a 51-letter sequence}) \\ \approx \frac{\# \text{ of sequences in which } w \text{ appears at least once}}{5000}.$$

The reason for using “at least once” is that the word may appear at multiple locations in the promoter, with only one occurrence at a particular location being sufficient for function.

The simulation provides an estimate of p_w that can be used with the normal approximation of the binomial with $n = 49$ trials and success probability p_w . Let N_w denote the number of promoter sequences in which w appears at least once. Then the statistic

$$Z_w = \frac{N_w - 49p_w}{\sqrt{49p_w(1 - p_w)}}$$

has approximately an $N(0, 1)$ distribution, which allows p-values to be computed for each word w . The results of this simulation are shown in Computational Example 3.7.

Computational Example 3.7: Number of promoter sequences containing at least one frequent k -word and statistical significance

Step 1: Compute the number of promoter sequences, N_w , containing each k -word

This time, we want to count the number of promoter sequences that contain at least one occurrence of each word. This quantity is computed by the plug-in in the function provided in Computational Example 3.6. The series of four “for” loops in the plug-in examines counts for all words found in sequence j and adds 1 to appropriate elements of `ncount` if a word appears, regardless of the number of times it appears. The desired counts of sequences N_w for any desired k -word `klmn` are extracted from the result of the previous computation, which is a list, as `prom.count$ncount[k,l,m,n]`. For example, the number of promoter sequences containing at least one instance of `AAAA` is

```
> prom.count$ncount[1,1,1,1]
[1] 13
```

We check the maximum value for N_w among all k -words.

```
> max(prom.count$ncount)
[1] 20 #Maximum value of Nw among all k-words
```

Step 2: Computation of p_w

This is done by simulation. We simulate 5000 sequences, each 51 nucleotides long and having the base composition of average *E. coli* DNA.

```
> ec.sim<-matrix(nrow=5000,ncol=51)
> for(i in 1:5000){
+ ec.sim[i,]<-sample(x,51,replace=T,prob.ec)
+ }
```

Remember that `x` is `[1,2,3,4]` and `prob.ec` is given in Computational Example 3.6. To get the data needed for p_w , we again apply the function `Ncount4()`:

```
> sim.count<-Ncount4(ec.sim,4)
```

This may take a while to run since there are nested loops operating on 5000 sequences. The values of p_w for the most abundant words listed in Table 3.2 are calculated as shown below for `AAAA`:

```
> sim.count$ncount[1,1,1,1]/5000
[1] 0.1238
```

The results for all words are presented in Table 3.3. Because the simulated sequences were based on chromosomal values for base frequencies, which are all nearly the same, the fraction of sequences for which each word appears should also be about the same.

Step 3: Computing p-values

We can use N_w and p_w computed above to calculate Z_w and then compute the desired p-value using the R function `pnorm` (previously used in Section 3.3). This is because Z_w is expected to follow (approximately) a normal distribution:

```
> Nw<-c(19,20,20,20,20,19,19,19,17,16) # See step 1
> pw
[1] 0.1238 0.1680 0.1710 ... 0.1660 0.1626 0.1736
```

(From Step 2 above.) We compute Z_w for the ten top-scoring k -words:

```
> options(digits=4)
> Zw<-(Nw-49*pw)/sqrt(49*pw*(1-pw))
> Zw
[1] 5.610 4.497 4.409 ... 4.172 3.497 2.826
```

Calculate the one-tailed p-value:

```
> 1-pnorm(Zw)
[1] 1.011e-08 3.452e-06 5.185e-06 2.328e-06
[5] 1.641e-06 1.589e-05 1.510e-05 1.510e-05
[9] 2.353e-04 2.354e-03
```

Table 3.3. Number of *E. coli* promoter sequences, N_w , containing indicated k -words for $k = 4$. Data for the ten most abundant words listed in Table 3.2 are shown. For the meaning of other quantities, see the text. The last column corresponds to p-values associated with each N_w . Entries not significant at level 0.001 (one-tailed test) are indicated in italics.

Word	N_w	p_w	Z_w	$\mathbb{P}(X > Z_w)$
TTTT	19	0.124	5.610	10^{-8}
CATT	20	0.168	4.497	0.000003
AATT	20	0.171	4.409	0.000005
TAAT	20	0.165	4.580	0.000002
ATTG	20	0.163	4.652	0.000002
TGAA	19	0.166	4.160	0.000016
ATAA	19	0.166	4.172	0.000015
ATTT	19	0.166	4.172	0.000015
TTTA	17	0.163	3.497	0.000235
ATTC	<i>16</i>	<i>0.174</i>	<i>2.826</i>	<i>0.002354</i>

Notice that N_w is lower than the number of overall occurrences shown in Table 3.2, as would be expected. From prior knowledge of *E. coli* promoters, we already expected that words contained within TATAAT would be abundant. Note that TAAT appears in about 40% of the listed promoters, as does ATAA. Testing for the occurrence and significance of TATA is left as an exercise.

Why did we earlier refer to this analysis as naive? This comes from the strong correlation between word counts, especially that which comes from word overlaps. If, for example, we knew that AAAA occurred in a particular string 49 times out of 51, then the end of the string of As must be a letter not equal to A. Of these, $p_T/(p_T + p_C + p_G)$ are expected to be Ts. That means that we expect at least $49 \times p_T/(p_T + p_C + p_G)$ occurrences of AAAT. In addition, there are occurrences of AAAT where the preceding letter was not an A. Taking all word overlaps into account in a rigorous statistical analysis is beyond the scope of this book, but you now know that N_w has different variances depending on w and that correlation between words is important.

Functional k -words are generally more complicated and more extensive than the example above, and in practice values to be used for k are not known in advance. Certainly $k = 4$ is too small; nevertheless, larger functional k -words can be decomposed into sets of characteristic 4-words. For promoters, $k = 6$ is a more realistic choice. In addition, our idea of counting exact occurrences does not correspond to what is observed in biological systems (e.g., not all promoters contain exact matches to the most frequent 4-words). However, it is often the case that exact word analysis has allowed researchers to make the initial pattern discovery. The approach above is not limited to exact word analysis, however. We could, for example, use $k = 6$ letter words and allow up to two mismatches in the definition of “occurrence.”

In Chapter 9, we illustrate how to describe signals of arbitrary length based upon patterns of letter occurrences observed among a set of aligned subsequences. The approach described in the current chapter could be extended to yield a complementary description of such signal sequences. We could implement this for the promoter data set by making histograms of positions at which each over-represented k -word occurs relative to the transcriptional start sites at +1. Any “signal” that appears within a window centered at position x relative to the transcriptional start site would then be represented by a word decomposition yielding the observed k -words preferentially mapping within that window (Galas et al., 1985).

References

Galas DJ, Eggert M, Waterman MS (1985) Rigorous pattern-recognition methods for DNA sequences. *Journal of Molecular Biology* 186:117–128.

Exercises

Exercise 1. Suppose that N has a Poisson distribution with parameter λ .

- Show that the probabilities in (3.2) sum to 1. [Hint: This uses the expansion of e^x that you learned in calculus.]
- Use the same expansion (by taking the first and second derivatives) to show $\mathbb{E}N = \lambda$ and $\text{Var}N = \lambda$.

Exercise 2. Verify that if U has the uniform distribution on (a, b) , then $\mathbb{E}U = (a + b)/2$ and $\text{Var}U = (b - a)^2/12$.

Exercise 3. Verify the formula (3.5) for calculating the variance.

Exercise 4. Use R to plot the probability density (3.7) of an exponential random variable for values of $\lambda = 0.5, 1.0, 5.0$.

Exercise 5. For the exponential distribution (3.7) calculate the mean and variance. [Hint: Use integration by parts.]

Exercise 6. For a distribution with probability density function $f(x) = \frac{3}{8}x^2$ for $0 \leq x \leq 2$ and $f(x) = 0$ elsewhere, find $\mathbb{P}(0 \leq X \leq 1)$, $\mathbb{E}X$, $\mathbb{E}X^2$, and $\text{Var}X$.

Exercise 7. Suppose Z has a standard normal distribution.

- Find $\mathbb{P}(-1 \leq Z \leq 1)$, $\mathbb{P}(-1 \leq Z \leq 2)$, $\mathbb{P}(-2 \leq Z \leq -1)$, $\mathbb{P}(-\infty \leq Z \leq 1)$.
- If $\mathbb{P}(Z \leq a) = 0.45$ and $\mathbb{P}(0 \leq Z \leq b) = 0.45$, find a and b . [Hint: Use `qnorm` in R.]

Exercise 8. In a certain genome the bases appear to be iid and $p_G = 0.3$. Define the (binomial) count of the number of Gs in the first 1000 bases as $N = X_1 + X_2 + \cdots + X_{1000}$.

- Give the mean and variance of N .
- Approximate, using the Central Limit Theorem, $\mathbb{P}(0 \leq N \leq 329)$ and $\mathbb{P}(285.5 \leq N \leq 329)$.
- Produce a histogram for 1000 replicates of N and compare the results with those of (b).

Exercise 9. A discrete random variable taking values $0, 1, \dots$ is said to have a *geometric* distribution with parameter p if

$$\mathbb{P}(N = k) = (1 - p)p^{k-1}, \quad k = 1, 2, \dots$$

- Suppose that X is exponential with parameter λ , and define a new random variable N by

$$N = k \text{ if } k - 1 < X \leq k.$$

Show that N is geometric, and identify p .

- b. Show that the mean $\mathbb{E}N$ of the geometric is $1/(1-p)$, and calculate the variance. [Hint: One way to do this is to complete the steps below:

$$\mathbb{E}N = \sum_{k=1}^{\infty} k(1-p)p^{k-1} = (1-p) \sum_{k=0}^{\infty} \frac{d}{dp} p^k = (1-p) \frac{d}{dp} \left(\sum_{k=0}^{\infty} p^k \right) = \dots .$$

For the variance, differentiate twice.]

- c. The geometric arises as the distribution of the number of tosses up to and including the first tail in a sequence of independent coin tosses in which the probability of a head is p . Use this to calculate the distribution of N . Can you derive $\mathbb{E}N$ using the waiting time analogy?

Exercise 10. Suppose N is binomial with $n = 1000$ and success probability p . Find a 90% confidence interval for p using $\hat{p} = N/1000$ when $N = 330$. What are $\mathbb{E}\hat{p}$ and $\text{Var}\hat{p}$?

Exercise 11. Assume that X_1, X_2, \dots, X_n are iid random variables having the exponential distribution with parameter λ . Find the maximum likelihood estimator of λ .

Exercise 12. Suppose $X = X_0, X_1, \dots, X_n$ are observations on a Markov chain with transition matrix $P = (p_{ij})$ and let $n(i, j)$ be the number of times that state i is followed by state j in the sequence X . Find the maximum likelihood estimator of the elements p_{ij} in terms of the $n(i, j)$. [Hint: See the discussion in Section 2.6.3.]

Exercise 13. Use the value of p computed for *HpaII* in Table 3.1. Compute $\lambda = 1/p$ for the parameter of the corresponding exponential distribution, using the approach of Section 3.3.

- For the bins $[0, 100)$, $[100, 200)$, $[200, 300)$, $[300, 400)$, $[400, 500)$, $[500, 600)$, $[600, \infty)$, compute the theoretical probability of each bin.
- Use the probabilities from (a) and the expected number of fragments from an *HpaII* digestion of bacteriophage lambda to calculate the expected number of fragments in each of the seven bins.
- Compute the X^2 value analogous to (2.29) on page 64 for these observed-expected data. The number of degrees of freedom for the approximate χ^2 distribution of X^2 is equal to $7 - 1 = 6$.
- Does the exponential distribution fit these data?

Physical Mapping of DNA

4.1 The Biological Problem

In contrast with genetic maps, physical maps of genomes or genome segments (e.g., chromosomal DNA) relate genome positions to each other using *physical distances* measured along the DNA helix axis. Distances between positions are often expressed in base pairs (bp) or multiples thereof (for example, kilobases (kb)— $\text{bp} \times 1000$). Large-scale physical maps are useful for some genome sequencing strategies, and they are even more important when studying genomes that have not been sequenced. However, sequencing “factories” can now churn out so much sequence per day that for small genomes it is easier and faster to determine the genome DNA sequence and identify the restriction sites computationally.

Markers on the physical map (discussed in detail in Section 13.4.2) allow investigators to retrieve particular regions of interest for further study. Restriction endonuclease (or restriction enzyme) cleavage sites represent one type of physical marker. *Genetic markers* (defined by mutations in genes) may affect the sizes of restriction fragments, leading to a correspondence between a genetic and a physical marker. Examples are mutations that destroy a restriction site (a restriction site polymorphism), or deletions, insertions, or inversions. Other types of markers are portions of a gene sequence. The presence of such markers can be detected by hybridization reactions, even if the complete DNA sequence is unknown. Sequences that are not part of genes can be used as physical markers, and these include segments of DNA containing variable numbers of tandem repeats and **sequence-tagged sites**, or **STSs**. The presence of a particular STS is revealed by the presence of PCR reaction products obtained by use of a particular primer pair so chosen that they allow amplification of DNA from only one location in a particular genome. Obviously, the ultimate physical map is the DNA sequence.

In Chapter 3, we discussed the properties of restriction digest products within the context of eventually constructing a restriction map. Recall that a restriction map is a display of positions on a DNA molecule that can be cleaved

by one or more restriction endonucleases and that this is one particular example of a physical map of a genome. As we will see below, reconstructing a restriction map from the restriction digest products can be computationally very hard. Laboratory workers circumvent this complexity by a number of experimental techniques, including use of multiple restriction enzymes, end labeling of DNA prior to digestion, analysis of incompletely digested end-labeled products, and hybridization. They also may employ genetic variants of the molecules being mapped (ones containing insertions, deletions, or inversions). Laboratory workers usually perform map construction by using a massively parallel, cranially mounted neural network (the brain). Optical mapping is an experimental strategy that preserves fragment order after digestion, but it requires nonstandard laboratory equipment in addition to more sophisticated statistical tools to allow for fragment length measurement errors and products of incomplete digestion.

As indicated in Chapter 3, DNA molecules can be too long to be handled conveniently without breakage. For this reason, genomes are intentionally fragmented into pieces of appropriate size for cloning into convenient cloning vectors. For example, pieces cloned into lambda vectors may be approximately 20 kb in length, those cloned into cosmids will be approximately 40 kb in length, and those cloned into BACs may be in the range of 100–300 kb. The inserts in these vectors can be individually mapped (the large-map problem is broken down into a large number of more tractable small-map problems). Then the genome map is built up by merging the maps of the constituent cloned inserts. One of the experimental issues is how many clones must be characterized before the genome map merges into a single continuous segment.

Physical maps can be informative for comparisons among organisms. For example, most unicellular eukaryotes and all multicellular animals and plants have organelles called mitochondria, which contain circular DNA molecules. The orders of the approximately 37 genes in the mitochondria of various animals have been determined, and they are not identical. A model to describe differences in order is based on breakage/rejoining or reversal of DNA segments, leading to permutations of the gene order. It should be possible to model this process in a manner that predicts animal phylogeny (branching pattern connecting lineages of organisms to common ancestors). This also has relevance when comparing bacterial strains with each other. For example, there are strains of *Escherichia coli* that are nonpathogenic and others that are pathogenic. There are species of *Yersinia* that give rise to epidemics of plague and others that do not. By comparing genomes (including gene orders), it is possible to reconstruct steps in the evolutionary history of these organisms that may provide insights into mechanisms of pathogenicity. We discuss permutations of gene order in the next chapter.

4.2 The Double-Digest Problem

This is an older, classical problem in computational biology, but we nevertheless describe it here to reinforce previous concepts and to provide experience with computationally complex problems. We will work with an idealized version of the problem that still illustrates the computational challenges.

4.2.1 Stating the Problem in Computational Terms

Given the sizes of restriction fragments from digestion of a DNA molecule by restriction endonucleases A, B, and a combination of those enzymes, A + B, reconstruct the order of restriction sites in the undigested molecule:

Products generated by digestion with A, sorted by increasing size,
 = $\{a_1, a_2, \dots, a_n\}$;
 Products generated by digestion with B, sorted by increasing size,
 = $\{b_1, b_2, \dots, b_m\}$;
 Products generated by digestion with A + B, sorted by increasing size,
 = $\{c_1, c_2, \dots, c_{m+n-1}\}$.

The solution is a list of positions for cleavage by A or B such that the fragment set $\{c_1, c_2, \dots, c_{m+n-1}\}$ is generated.

4.2.2 Generating the Data

A sample of DNA is “digested” with a restriction enzyme until all sites on every molecule are cut. Agarose gel electrophoresis gives a way to separate DNA fragments (which are negatively charged) by their lengths. The molecules migrate through the porous agarose a distance proportional to the negative logarithm of their lengths; big pieces hardly move, and small pieces fall like stones. This principle also allows DNA to be sequenced, a topic we treat later. The approximate fragment lengths are determined and listed in order of size. The problem is to reproduce the order of the pieces in the original DNA. That is, gel electrophoresis gives us the set of lengths unordered relative to the actual location in the DNA. For example, digesting phage lambda DNA (48,502 bp) with *Eco*RI gives the following lengths (in kb): 3.5, 4.9, 5.6, 5.8, 7.4, and 21.2. The correct order on the lambda genome is 21.2–4.9–5.6–7.4–5.8–3.5.

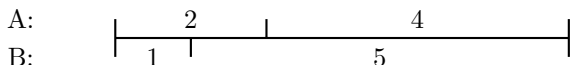
If the molecule has been sequenced, we can also learn this ordering by downloading the sequence from a database and searching the string for occurrences of GAATTC. How are restriction maps of uncharacterized DNA molecules determined experimentally?

4.2.3 Computational Analysis of Double Digests

As we indicated above, one experimental approach to the restriction mapping problem is to select another enzyme, digest with that enzyme, and finally digest with a combination of both enzymes. Inferring the two-enzyme map of the DNA from the resulting data is called the **double-digest problem (DDP)**. Here is a simple example for a linear molecule digested with enzymes A and B (in these examples, fragment sizes are taken to be integers):

Products produced from digestion with A : {2, 4};
 Products produced from digestion with B : {1, 5};
 Products produced from digestion with A + B : {1, 1, 4}.

This “toy” problem can be solved by inspection. The “ends” in the double digest must be 1 and 4. This is because the shorter of the fragments produced at each end by digestion with A or B must lie wholly within the larger of the end fragments produced by A or B. This observation solves the problem. The solution is



In general, there are $(2!)(2!) = 4$ different potential maps or orderings for this problem (two orderings for A fragments and two for B fragments), and each ordering has a biologically indistinguishable reversal. That is,

A : 2 4 and A : 4 2
 B : 1 5 B : 5 1

are reversals of each other. They correspond to reading fragment orders from one end of the molecule or the other.

It is easy for these problems to become more complex:

Products produced from digestion with A : {3, 4, 5};
 Products produced from digestion with B : {2, 4, 6};
 Products produced from digestion with A + B : {1, 1, 2, 3, 5}.

Obviously, 5 must be at the end, and therefore 6 must be also. This is because there must be a fragment at each end from the A or B digests that is not cleaved after double digestion. Fragment 5 must lie within 6 since 6 is the only B fragment large enough to contain it. Since in this problem only integer values are allowed, one end of 5 must match one end of 6, which means that 5 is an end fragment from the A digest and 6 is an end fragment from the B digest:

(two fragments)–5
 (two fragments)–6

Having determined the two end fragments, we have only four possibilities left to consider:

A : 3 4 5 or 3 4 5 or 4 3 5 or 4 3 5
 B : 2 4 6 or 4 2 6 or 2 4 6 or 4 2 6

The last one is eliminated because it would produce too few double-digest pieces. The techniques used in these two examples obviously depend on the small sizes of the problems.

A more systematic method to predict the double-digest products is as follows: (1) take the fragment lengths, (2) produce maps of *positions*, (3) interleave the positions, and (4) take successive differences. For example, the digests above correspond to a molecule that is of length 12. The first map position is taken as 0, and the last position is 12. Starting at 0, successive positions on the map are produced by adding the lengths of fragments, starting from the left end:

Fragment ordering	→	Map positions	→	Interleaved positions
3 4 5		0 3 7 12		0 2 3 6 7 12
2 4 6		0 2 6 12		

Successive differences:

$$2 - 0, 3 - 2, 6 - 3, 7 - 6, 12 - 7 \quad \longrightarrow \quad 2, 1, 3, 1, 5$$

(This is a solution because the correct double-digest products are produced.)

Now we try another ordering:

Fragment ordering	→	Map positions	→	Interleaved positions
3 4 5		0 3 7 12		0 3 4 6 7 12
4 2 6		0 4 6 12		

Successive differences:

$$3 - 0, 4 - 3, 6 - 4, 7 - 6, 12 - 7 \quad \longrightarrow \quad 3, 1, 2, 1, 5$$

(This is another solution!)

Now try the putative map positions of one of the other possibilities:

Fragment ordering	→	Map positions	→	Interleaved positions
4 3 5		0 4 7 12		0 2 4 6 7 12
2 4 6		0 2 6 12		

Successive differences:

$$2 - 0, 4 - 2, 6 - 4, 7 - 6, 12 - 7 \quad \longrightarrow \quad 2, 2, 2, 1, 5$$

This does not correspond to the observed products of double digestion, so this pair of fragment orders is rejected. The remaining possible ordering also fails to agree with the double-digest data, as you can verify for yourself.

The solutions to the double-digest problem are:

A : 3 4 5 and its reversal 5 4 3
 B : 2 4 6 6 4 2

and

A : 3 4 5 and its reversal 5 4 3
 B : 4 2 6 6 2 4

4.2.4 What Did We Just Do?

Basically, we decided to try all orderings of each digestion product and to find out whether the predicted double digest fits the data. Here is an outline of the method. We assume that no cut site for A coincides with a cut site for B.

- (1) Input fragment sizes:
 a_1, a_2, \dots, a_n ($n - 1$ cuts),
 b_1, b_2, \dots, b_m ($m - 1$ cuts),
 $c_1, c_2, \dots, c_{m+n-1}$ ($m + n - 2$ cuts).
- (2) Produce position maps:
 $0, a_1, a_1 + a_2, \dots, a_1 + a_2 + \dots + a_n = L,$
 $0, b_1, b_1 + b_2, \dots, b_1 + b_2 + \dots + b_m = L.$
- (3) Merge and put into order.
- (4) Then take differences of adjacent numbers in (3).
- (5) Check whether output of (4) is equal to $c_1, c_2, \dots, c_{m+n-1}$.

When we solved our first tiny problems, we used special tricks for the data we had. Then we created a general approach to DDP that is correct and systematic.

What is the space efficiency? By this we mean memory commitments in our computer. For each map check, we needed space proportional to $n + m$. In addition, we need to save all correct permutations, and that we cannot estimate. The map check requires space proportional to $(n + m)$.

Our examples were all small problems. How complicated can things get with this approach? That is, what is the time efficiency? The number of possible orderings of the A fragments is $n!$, and the number of possible orderings of the B fragments is $m!$, and so there are $n! \times m!$ different combinations of position maps in (2) that require testing. As n and m become larger, the number of possibilities to test becomes huge!

n	1	2	4	6	8	10
$n!$	1	2	24	720	40,320	3,628,800

This means that a problem with ten A fragments and ten B fragments would yield over 10^{13} subproblems (2) through (4) to be analyzed. A very simple data set becomes computationally large very fast.

4.3 Algorithms

We now turn to specifying a method for solving computational problems such as DDP. The description of such a specification is called an algorithm. It should be kept in mind that this is a slightly higher level than computer code, although algorithms can and should result in code.

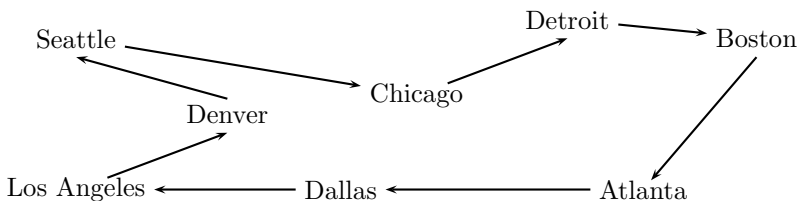
First of all, we need a set of basic operations such as addition, multiplication, etc. Next we describe some important aspects of algorithms (see Skiena, 1998).

- (1) The input to and output from an algorithm must be precisely specified. (In the DDP, the input is three unordered lists (of integers), each of which must sum to the same number. The output is all orderings of the first two lists that solve the DDP.)
- (2) The algorithm must consist of a well-defined series of operations.
- (3) The algorithm must stop after a finite number of steps.

Finally, we must consider the notions of *correctness* and *efficiency*. Correctness is a mathematical issue of proving that the algorithm does what it should do. The DDP as we set it up takes a multiple of $n! \times m! \times (n + m)$ operations. This is an example of the time efficiency or time *complexity* of an algorithm. The computational time complexity of an algorithm with input data of size n is measured in “big O ” notation and is written as $O(g(n))$ if the algorithm can be executed in time (or number of steps) less than or equal to $Cg(n)$ for some constant C . For example, adding the numbers a_1, a_2, \dots, a_n can be accomplished in time $O(n)$ by the following method. Set $S = 0$. Then for $j = 1, 2, \dots$ set $S = S + a_j$. On step n , S equals the desired sum. More generally, an algorithm has polynomial time complexity if $g(n)$ is a polynomial, and if $g(n) = a \times b^n$ for $a > 0$ and $b > 1$, the algorithm has exponential time complexity.

Our method for DDP is correct but hardly efficient. Can we find a more efficient method that is also correct, say one that takes at most a multiple of nm or even $n + m$ steps? The answer is very likely to be *no*. Here is the reason.

There is a famous problem called the **traveling salesman problem (TSP)**. Cities $1, 2, \dots, n$ are located with various flights connecting them. The TSP is to find an ordering of cities $i_1 \rightarrow i_2 \cdots \rightarrow i_n \rightarrow i_1$ so that all cities are visited and the total route length is minimized. An example is shown below:



If the salesman starts in Seattle and follows the itinerary shown, is the trip shown shorter than if he were to end the trip by going from Dallas to Denver and then to Los Angeles and Seattle? This problem (in general) is one of the so-called NP-complete problems that have equivalent computational difficulty. It is unknown whether there is a polynomial method for solving these problems, but the universal belief is that there is no such method. The DDP is one of these problems without a known polynomial algorithm.

One more aspect of studying and implementing algorithms is space efficiency. As described above, our DDP algorithm has time efficiency $O(n! m! (n+m))$ (Problem 6). Space efficiency is measured in the same “big O ” notation. We remark that in Section 4.2.4 there was an algorithm for taking fragment permutations and producing the double digest lengths. That algorithm can be implemented in $O(n+m)$ time and space. We have not described how to generate all $n! m!$ pairs of permutations!

4.4 Experimental Approaches to Restriction Mapping

When applying computational methods to biological problems, sometimes it is better to invest more effort in computational approaches, and other times it is easier to generate a different data set in the laboratory. We just showed that the double-digest problem rapidly becomes extremely complicated even for modest numbers of fragments. Even a small microbial genome containing, for example, 1000 *EcoRI* fragments and 1000 *HindIII* fragments, if analyzed as described above, would require analysis of an astronomically large number of subproblems. This effort can be avoided by using a different experimental approach.

In the examples above, molecules were digested to completion, so that all fragment orderings were completely disrupted. There are two experimental approaches that retain ordering information. *Optical mapping* (Cai et al., 1995) involves stretching individual DNA molecules onto a solid substrate, digesting them on the substrate, and then observing the ordered digestion products by fluorescence microscopy. Lengths of the product fragments (relative to included size standards) can be measured from recorded images in the microscope field, and lengths can also be determined from fluorescence intensities. This is particularly useful when employed with restriction endonucleases such as *NotI* that generate very large fragments.

Another approach that preserves order information employs intentional incomplete digestion, as illustrated in Fig. 4.1. At first consideration, this might seem to make analysis worse because if n fragments are produced by complete digestion, the total number of fragments produced by incomplete digestion of a linear molecule is (Problem 8)

$$\binom{n+1}{2}.$$

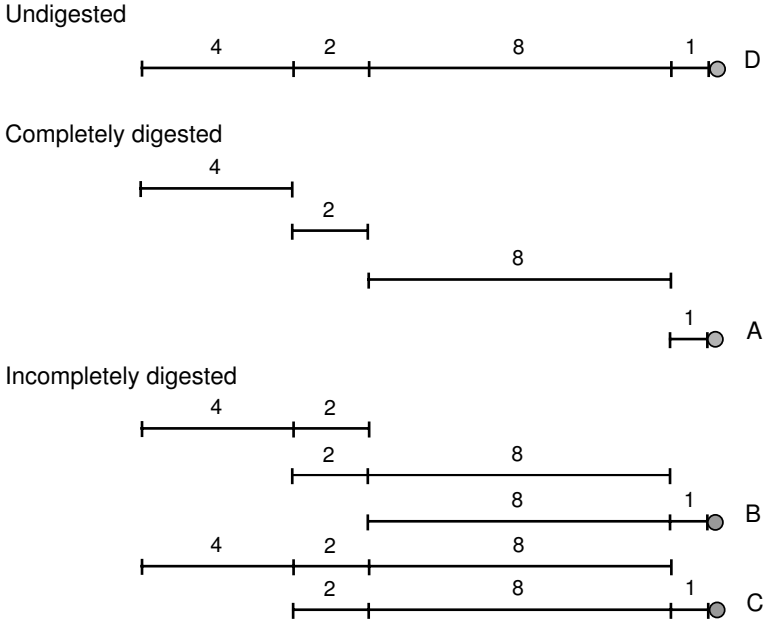


Fig. 4.1. Using incomplete digestion as an aid to restriction mapping. The molecule to be mapped is labeled at one end (filled circle) with a radioisotope. By autoradiography or the use of a phosphorimager, only radioactively labeled fragments (A, B, C, and D) will be detected after resolution by gel electrophoresis. The size increments separating each successive fragment starting with the end fragment correspond to the distance to the next restriction site.

The experimental “trick” (Smith and Birnstiel, 1976) is to radioactively label (directly or indirectly) one end of the molecule. In Fig. 4.1, the filled circle marks the labeled end fragment.

The mixture of completely and incompletely digested products can be resolved electrophoretically, and the sizes of the radioactive fragments can be measured (for example, after autoradiography of the dried gel or using a Southern blot, or with the help of a phosphorimager). Scoring only the labeled fragments (only they appear on the autoradiogram) eliminates the background of incomplete digestion products that do not include the indicated end fragment. Now we list the labeled fragments in order of size:

0 1 9 11 15

This includes the map of restriction site positions measured from the right end!

In what was then an experimental tour de force, Kohara et al. (1987) applied this approach for constructing a restriction map of the *E. coli* genome for several different restriction enzymes. They mapped clones in a lambda

library and then assembled the larger map by overlap. This general process, which in some form is used in most genome analyses, will be discussed in the next section. In the case of Kohara et al., individual lambda clones were indirectly end-labeled by hybridizing radioactive end probes to incomplete digests of cloned DNA that had been transferred to membranes by Southern blotting.

4.5 Building Contigs from Cloned Genome Fragments

4.5.1 How Many Clones Are Needed?

We indicated previously that analysis of genomes often proceeds by first breaking them into pieces. This is because genomes of free-living (nonviral) organisms are much larger than the sizes of inserts in most cloning vectors. For example, microbial genomes usually have genome sizes $G > 0.5 \times 10^6$ bp, and for mammalian genomes, $G > 10^9$ bp are usual. In contrast, capacities of cloning vectors are on the order of 10^4 bp for lambda or cosmid vectors and 10^5 to 10^6 bp for BAC and YAC vectors, respectively. This means that genomes will be represented by **genomic libraries**, which are collections of clones that contain a given genome represented piecewise as inserts in a specified cloning vector.

How many clones N should there be in the library? This depends upon the following parameters:

G = genome length in bp,

L = length of the clone insert in bp,

f = probability that any chosen base pair is represented in the library.

Suppose that we were to select one clone. The probability that a particular base *is* recovered in that clone is just the fraction of the genome contained in the clone, L/G . The probability that any particular base *is not* in the clone (i.e., is not “covered” by the clone) is

$$\mathbb{P}(\text{not covered, one clone}) = 1 - L/G. \quad (4.1)$$

The probability that any particular base is not covered after N independent clones have been selected is therefore

$$\mathbb{P}(\text{not covered, } N \text{ clones}) = (1 - L/G)^N. \quad (4.2)$$

If f_c is the probability that a base is covered after N clones have been drawn, then

$$1 - f_c = \mathbb{P}(\text{not covered, } N \text{ clones}) = (1 - L/G)^N. \quad (4.3)$$

We can solve (4.3) for N (after taking the logarithm of both sides) to yield an expression for estimating the number of clones needed in the library in terms of the probability f_c that any base will be covered (e.g., $f_c = 0.95$):

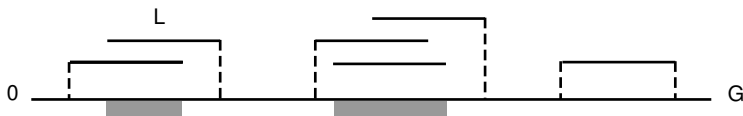


Fig. 4.2. Coverage of a genome of length G represented by a genomic library, with inserts of length L for each clone. Some areas of the genome (shaded) may be represented multiple times in the library, whereas others are represented once or not at all. The **coverage** is the *average* number of times that each position in the genome is represented in the library. Broken lines indicate limits of the three contigs (gap-free assemblies of genome segments represented by two or more clones) shown.

$$N = \log(1 - f_c) / \log(1 - L/G). \quad (4.4)$$

Computational Example 4.1: How many clones?

How many cosmid clones are required in a library representing the *E. coli* genome such that $f_c = 0.95$?

We know that $G = 4.6 \times 10^6$ bp and $L = 4 \times 10^4$ (for cosmids). Hence, from (4.4), $N = \log(1 - 0.95) / \log(1 - 4 \times 10^4 / 4.6 \times 10^6) = 343$ clones.

From Computational Example 4.1, we can calculate that the amount of DNA represented in the clones is $343 \text{ clones} \times 4 \times 10^4 \text{ bp/clone} = 14 \times 10^6$ bp. This is about three times the size of the *E. coli* genome. Some portions of the genome are represented multiple times and some not at all, but, on average, any position will be represented (covered) approximately three times in this particular library (see Fig. 4.2 for an explanation).

The **coverage** (the number of times, on average, that any base pair b is contained in inserts belonging to members of the library) is defined as

$$c = NL/G. \quad (4.5)$$

Equation (4.3) can be rewritten in terms of coverage:

$$1 - f_c = (1 - L/G)^N = (1 - (NL/G)/N)^N = (1 - c/N)^N \approx e^{-c}. \quad (4.6)$$

Thus we find that

$$f_c = 1 - e^{-c}. \quad (4.7)$$

The amount of coverage required to achieve different probabilities of including any particular base pair is

c	1	2	3	4	5
f_c	0.632	0.865	0.950	0.982	0.993

From this, we see that if we want to have a 99% chance of having a locus of interest in our library, we will need to clone five genome equivalents.

From Exercise 7, we will discover that nearly half a million lambda clones would be required to represent mammalian genomes with a reasonable degree of completeness, and from the table above, we see that half again as many clones would be needed for f_c to be 0.99. Actually handling libraries of this size (e.g., propagating clones, distributing clones, arraying clones on hybridization filters) is very labor-intensive, requiring robotics for manipulating the clones and databases for recording clone inventory and annotation.

As we randomly select more and more clones, we do eventually begin to fill in the gaps shown in Fig. 4.2, but at late stages of this process, each successive clone, picked at random, is far more likely to fall into a region already covered than it is to fall into a region of the genome that has not been covered at all. At that point, a better approach is to switch to a directed experimental strategy for *gap closure*—the process of covering regions not previously covered.

4.5.2 Building Restriction Maps from Mapped Clones

As is the case for complete digestion with restriction enzymes, the cloning process disrupts the ordering of the cloned segments. The inserts are generated either by shearing the DNA or by restriction digestion, and usually the inserts are size-selected before ligation to the vector sequences. Size selection is needed particularly for vectors that accept inserts in a particular size range. For example, lambda vectors accept inserts ranging from 9 kb to 20 kb, and cosmid vectors require inserts of length $L \sim 40$ kb. When individual clones are selected, their positions on the genome are unknown, and we must build up the complete restriction map from pieces of it. Developing a larger segment of the map from data for smaller segments is called the “bottom-up” approach, which we will discuss in greater detail in Chapter 8.

Using techniques such as the incomplete digestion method (Fig. 4.1), the restriction map of individual clones can be determined. If the inserts in two different clones contain some of the same restriction sites (i.e., generate restriction fragments of the same length), then the inserts may share a region in common (they overlap), and a larger, contiguous mapped region can be recognized by extending the physical map on both sides of the region of overlap (Fig. 4.3A). A **contig** is a genome segment represented by two or more overlapping clones. A contig is part of a larger physical map, and its material representation is a particular set of clones. If the genomic library is very large, nearly all of the clonable portions (heterochromatin is difficult to clone) will be represented by clone inserts, and most regions will be represented many times on different clones, as shown in Fig. 4.3C.

The desired outcome is to place each clone in its correct genome location, as shown in Fig. 4.3C. A typical step in the process for achieving that outcome is illustrated in Fig. 4.3A. The question is, “How rapidly will the map approach completion as coverage c increases?” When should we stop picking clones at random and move to directed gap closure? These questions can be answered using Poisson statistics, which were introduced in Chapter 2.

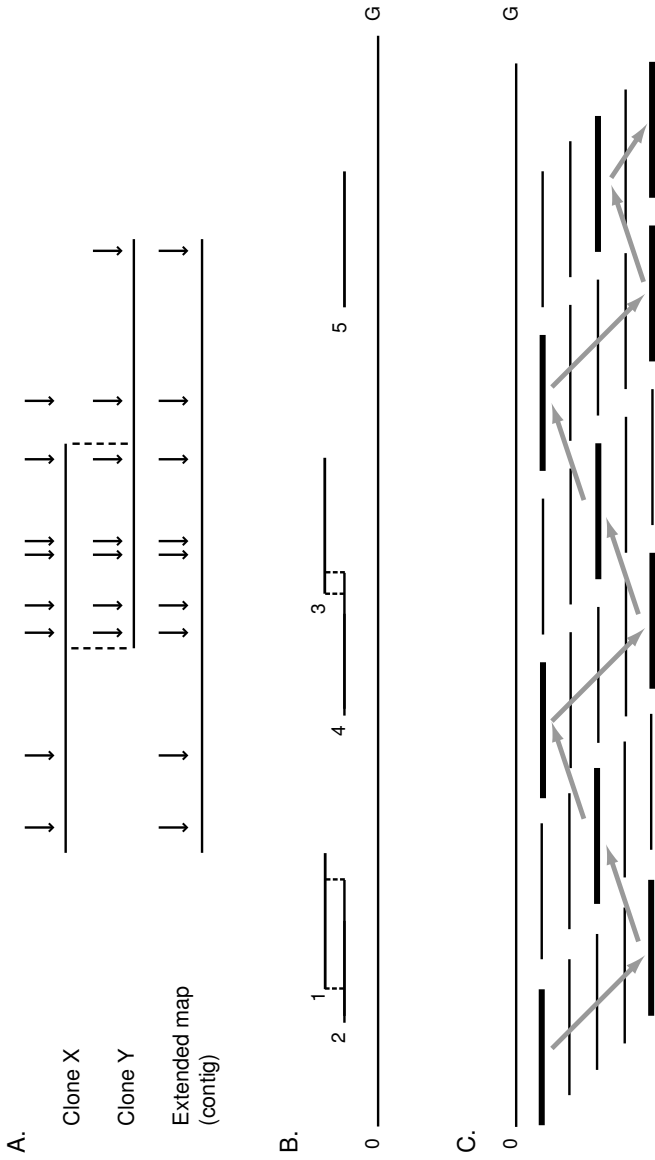


Fig. 4.3. Mapping a large genome by assembling maps of representative clones. Panel A: Clones X and Y are seen to overlap because they share some of the same restriction fragments. This allows their maps to be joined to form a **contig**. Panel B: Cloned inserts can overlap sufficiently for that overlap to be detected (1 and 2), may overlap insufficiently for overlap detection (3 and 4), or may be singletons (5) having no context information supplied by neighbors. Contigs, singletons, and selected clones with undetected overlaps are called **islands** (see the text). Panel C: Once multiple clone overlaps have been constructed such that they span a whole genome or genome segment, there will be several clones covering each region. In this illustration, coverage $c \sim 5$. To reduce the redundancy, a subset of minimally overlapping clones is chosen (heavy lines) that still spans the whole genome. The path through the graph representing these clones (arrows) is called a *minimal tiling path*.

4.5.3 Progress in Contig Assembly

Figure 4.3B illustrates the possible situations encountered for assembly of contigs. The process is dependent on recognizing overlaps (Fig. 4.3A), and it should be evident that some overlaps are too short to recognize (Fig. 4.3B, 3 and 4). This could occur if the overlap is shorter than the average fragment size generated by the restriction endonuclease employed in mapping, for example. Also, because of the experimental error in measuring restriction fragment lengths, identical fragments might not be recognized as such. Moreover, it is possible to generate similar-sized fragments from inserts from totally different parts of the genome. For this reason, it is usual to demand size matches among several fragments from each clone before declaring that two clones overlap.

In Fig. 4.3B, clones 1 and 2 represent a contig that can be recognized, while clones 3 and 4 represent a contig that cannot be recognized. Clone 5 is a singleton. These three blocks of sequence (two contigs and a singleton) correspond to fixed points in the genome in a “sea” of genomic DNA. Contigs (those that are apparent and those not recognized) together with singletons constitute “islands.”

To analyze progress in contig assembly, we list again the parameters pertaining to the problem (some of which are identical to the ones used in Section 4.5.1):

- N = number of clones;
- L = length of insert in each clone;
- G = genome length;
- Ω = minimum amount of overlap required for detection of that overlap;
- θ = fraction of L corresponding to Ω ; $\Omega = \theta L$.

Overlaps can be detected only if they exceed $\Omega = \theta \times L$. Smaller overlaps are undetected by the experimenter. As before, the coverage is $c = NL/G$.

Now we develop expressions for the number of islands observed for a given value of c . We start by examining the distributions of clone inserts along the genome. (Any island must start with one particular clone.) Of course, now we are distributing intervals along the genome instead of positions. We can use our previous formalism if we locate each interval (cloned insert) by specifying the position of one end. We will pick the left end (we could have chosen the right), and once the position of the left end on the genome is specified, the remaining $L-1$ base pairs are located to the right of this position in succession.

When analyzing restriction sites, we didn’t know at first how many of them there were, and we had to calculate λ , the probability that any position on the genome is the beginning of a restriction site. But now we know that N clones have been drawn, so we can say immediately that the probability that any genome position corresponds to the left end of a clone is $\lambda = N/G = c/L$ (i.e., in G base pairs the expected number of clone ends is $\mathbb{E}(Y_N) = G\lambda$). Thus we have “converted” a known number (N) of clones into a random variable with expectation M . According to the Poisson distribution, for example,

$$\begin{aligned} \mathbb{P}(\text{Number of left ends in an interval of length } x = k) \\ = \frac{(x\lambda)^k}{k!} e^{-x\lambda} = \frac{(xc/L)^k}{k!} e^{-xc/L}, \end{aligned} \quad (4.8)$$

where x is the length of the interval along a DNA segment.

Now let's think about the number of apparent islands that will be observed. We have to use the word "apparent" because some clone pairs will actually overlap but not sufficiently to detect—they will be counted as if they were two islands instead of one. We will enumerate the number of islands by counting their right ends. This is the same as the number of clones that are at the right end of an island. Let Γ be the expected number of islands. Then

$$\Gamma = N\mathbb{P}(\text{A clone is at the right end of an island}). \quad (4.9)$$

Our problem is solved if we can calculate the probability that a clone is at the right end of an island. We can see what this probability is if we consider the diagram of a particular clone insert shown below (let's name it 8G11—the clone in row G, column 11 of microtitre plate 8):

$$8G11 \quad \underbrace{\hspace{10em}}_{(1-\theta)L} \quad | \quad \theta L$$

What is the probability that clone 8G11 is at the right end of an island? If there is another clone whose left end lies within θL of the right end of clone 8G11, we can't tell whether there are any clones to the right of it. However, if the left end of a clone does lie within the segment of length $(1-\theta)L$ diagrammed above, we can be sure that clone 8G11 is not at the right end of an island. The probability that the left end of another clone does not lie in the interval of length $(1-\theta)L$ (i.e., the probability that 8G11 is at the right end of a contig) is given by (4.8), with $k = 0$ and $x = (1-\theta)L$. The expected number of islands then becomes

$$\Gamma = Ne^{-(1-\theta)LN/G} = Ne^{-(1-\theta)c}. \quad (4.10)$$

To obtain an expression convenient for plotting, multiply and divide the expression on the right by L/G to obtain

$$\Gamma = (G/L)(ce^{-(1-\theta)c}). \quad (4.11)$$

This result is the number of islands expected as a function of coverage and θ for given values of G and L .

The results for different values of θ are plotted in Fig. 4.4. The verbal explanation for the shapes of the curves is as follows. When the mapping is started, there are no islands. At the beginning, as we draw each new clone for mapping, it is more likely to be a singleton (at that stage) than to overlap a clone already drawn, so the number of islands begins to climb. As the number of clones examined increases, existing contigs begin to increase in length as clones overlap their ends. So, later in the process, an ever-increasing proportion of the genome is being spanned by contigs. Any new clones that fall inside

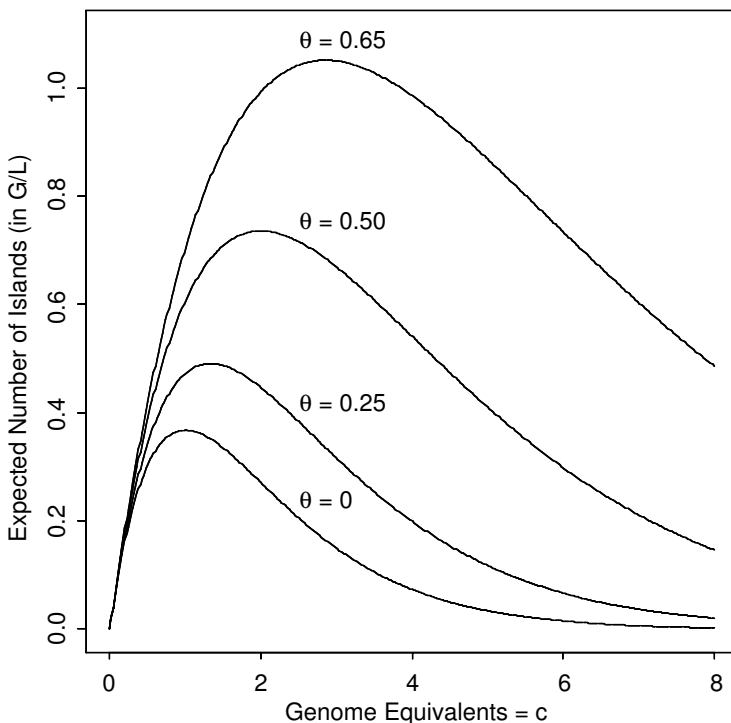


Fig. 4.4. Expected number of islands as a function of coverage for different values of fractional overlap, θ , required to reliably detect that overlap.

an existing contig do not add to the number of contigs or islands. Some clones will overlap the left end of one contig and the right end of another, causing the contigs to merge. This means that the number of contigs and islands begins to drop later in the process. Eventually, the number of islands is expected to approach the final value of $I = 1$ (all clones merged into the single contig representing the entire genome). But the approach to $I = 1$ gets slower and slower as c increases. This is because, at the later stages of the process, most of the genome has been spanned, and the probability that a random clone will fall in the small gaps between the large contigs gets smaller and smaller.

We can estimate the number of “singletons,” clone inserts not overlapping others, by using the same argument. A clone insert C is a singleton if there is no other clone whose left end falls in the interval of length $(1 - \theta)L$ at the left end of C (probability = $e^{-(1-\theta)c}$) and there is no other clone whose right end falls in the interval of length $(1 - \theta)L$ to the right of C . This has the same probability as for the left end, so the joint probability that there is no other clone whose left end falls in the intervals $(1 - \theta)L$ to the left and to the right of C is $e^{-2(1-\theta)c}$ (the product of the two identical probabilities). The predicted number of singletons is then

$$\# \text{ of singletons} = Ne^{-2(1-\theta)c}. \quad (4.12)$$

Computational Example 4.2: Reality check

Kohara et al. (1987) produced a complete restriction map for *E. coli* by using a random clone-mapping strategy employing inserts contained in lambda cloning vectors. The parameters were

$$N = 1025, \quad L = 1.55 \times 10^4, \quad G = 4.7 \times 10^6.$$

The problem is to calculate the numbers of islands and singletons and compare them with the experimental results.

Overlap was detected by analysis of restriction sites for eight different enzymes mapped in each of the clone inserts. Six consecutive sites were required to declare overlaps. Since they calculated that the eight enzymes that they used would cut the genome in 9700 locations, the average number of cleavage sites in any one 15.5 kb insert would have been about 32.

Calculated results

$$c = NL/G = 3.38; \quad \theta = 6/32 \approx 0.19.$$

From (4.11), we calculate

$$\Gamma = (303)(3.38)(e^{-2.84}) = 66.3 \text{ islands.}$$

From (4.12), we calculate

$$\# \text{ of singletons} = (1025) \times e^{-2(1-0.19)(3.38)} = 4.3.$$

Experimental results

Kohara et al. (1987) reported 70 islands, seven of which were singletons. The agreement between theory and experiment is excellent given the statistical sampling of clones. Olson et al (1986) reported another early example of this restriction mapping strategy.

4.6 Minimal Tiling Clone Sets and Fingerprinting

For some purposes (e.g., various shotgun sequencing strategies—see Chapter 8), it is not necessary to generate a complete restriction map. Instead, we seek a representation of the genome as a set of ordered clones that minimally overlap and yet include all of the genome sequence. Such a set of clones is called a **minimal tiling clone set** (Fig. 4.3C). It is certainly possible to produce such a set from a large library whose inserts have all been mapped, but this is not needed in general.

One way to produce a minimal tiling set is by fingerprinting the clones (Wong et al., 1997). A **fingerprint** of a clone insert is the set of fragments or fragment sizes produced from digestion by one or more restriction enzymes. Different inserts will produce different fingerprints. As we indicated above, shared fragments from mapped inserts can be used to build contigs. With fingerprinting, the mapping step is skipped, and overlaps between two clones are declared when they share the requisite number of fragments whose sizes match within experimental error. With this procedure, restriction fragments are “binned” and partially ordered, even though the clones have not been completely mapped.

For example, suppose a digest of clones A, B, and C by the same restriction enzyme or combination of enzymes yields the fragment clone fingerprints indicated below. (Fragment sizes in kb are listed in order of size, and insert sizes are compatible with typical lambda cloning vectors. Assume that enzyme cleavages occur at the boundaries between the insert and vector sequences and that vector fragments have been removed from the list.)

A	B	C
7.0	6.0	6.0
5.0	4.0	5.0
3.5	3.0	4.0
3.0	2.5	3.0
2.0	2.0	2.0
1.0	1.5	1.0
	1.0	

Notice that A and C share fragments of sizes 5.0, 3.0, 2.0, and 1.0 kb. C and B share fragments of sizes 6.0, 4.0, 3.0, 2.0, and 1.0. If four matching fragments suffice to declare overlaps, then the following sets of fragment clusters are defined:

A : (7.0, 3.5) (5.0, 3.0, 2.0, 1.0)
 C : (5.0, 3.0, 2.0, 1.0)(6.0, 4.0)
 B : (6.0, 4.0, 3.0, 2.0, 1.0)(2.5, 1.5)
 C : (5.0) (6.0, 4.0, 3.0, 2.0, 1.0)

We don't know the order of the fragments enclosed in parentheses: we have arbitrarily listed them in order of decreasing size. But we have defined groups of fragments that occur as neighbors (in some undetermined order) in the restriction map. Since A and B both overlap C and share fragments 3.0, 2.0, and 1.0, we see that we can overlap all three clones. The particular clusters defined by each pair of clones allow us to further reduce the sizes of unmapped clusters:

A : (7.0, 3.5) 5.0 (3.0, 2.0, 1.0)
 C : 5.0 (3.0, 2.0, 1.0) (6.0, 4.0)
 B : (3.0, 2.0, 1.0) (6.0, 4.0) (2.5, 1.5)

The result is a partial restriction map of the region represented on these clones, and this map was obtained by using the clone fingerprints *without mapping the individual clone inserts*:

(7.0, 3.5) 5.0 (3.0, 2.0, 1.0) (6.0, 4.0) (2.5, 1.5)

Given these clones and the mapping result, the region could be represented by clones A and B only. They would be the minimal tiling clone set for this region:

A : (7.0, 3.5) 5.0 (3.0, 2.0, 1.0)
 B : (3.0, 2.0, 1.0) (6.0, 4.0) (2.5, 1.5)

In this toy example, we had only three clones to compare. In actuality, there is experimental error in determining fragment sizes, and the number of clones in the library may be very large. This could make the fingerprinting process tedious because there would be $N(N - 1)/2$ fingerprint comparisons to be made. If the library contained 100,000 clones, this would involve about 5×10^9 comparisons in all.

It is possible to reduce the labor of producing a minimal tiling clone set by prescreening the library for clones that overlap. We will describe this in more detail in Chapter 8, but we note here that it is relatively easy to determine the sequences for about 500 bp into the insert from each end of the segment cloned. With this sequence, we can design PCR primers that will amplify DNA between them wherever that sequence is present and, in particular, within the inserts in overlapping clones. We would then start with clone X and design primers for its “right” end. By using appropriate pooling schemes, it is relatively easy to identify other clones that overlap the right end of X because an amplified product can be produced from the appropriate clone pools. Those are the only clones that need to be fingerprinted to determine the ones that have minimal overlap with the right end of X. If the left end of Y overlaps the right end of X, then we can design primers for the *right end* of Y, screen the library for clones that overlap its right end, fingerprint *those clones*, and continue the process until we have produced a minimal tiling path clone set. What we have just described is equivalent to employing **sequence-tagged connectors** to generate a minimal tiling clone set like those used for clone-by-clone shotgun sequencing (Chapter 8).

References

Cai W, Aburatani H, Stanton V Jr, Housman DE, Wang Y-K, Schwartz DC (1995) Ordered restriction endonuclease maps of yeast artificial chromo-

- somes created by optical mapping on surfaces. *Proceedings of the National Academy of Sciences USA* 92:5164–5168.
- Kohara Y, Akiyama K, Isono K (1987) The physical map of the whole *E. coli* chromosome: Application of a new strategy for rapid analysis and sorting of a large genomic library. *Cell* 50:495–508.
- Olson MV, Dutchik JE, Graham MY, Brodeur GM, Helms C, Frank M, MacCollin M, Scheinman R, Frank T (1986) Random-clone strategy for genomic restriction mapping in yeast. *Proceedings of the National Academy of Sciences USA* 83:7826–7830.
- Skiena S (1998) *The Algorithm Design Manual*. New York:Springer-Verlag.
- Smith HO, Birnstiel ML (1976) A simple method for DNA restriction site mapping. *Nucleic Acids Research* 3:2387–2398.
- Wong GKS, Yu J, Thayer EC, Olson MV (1997) Multiple-complete digest restriction fragment mapping: Generating sequence-ready maps for large-scale DNA sequencing. *Proceedings of the National Academy of Sciences USA* 94:5225–5230.

Exercises

Exercise 1. Given the products of digestion with A, B, and A + B shown below, determine the restriction map(s). Assume that the molecule is linear and that the sizes of digestion products are integers.

- Products produced from digestion with A : {8, 9, 10};
- Products produced from digestion with B : {4, 5, 8, 10};
- Products produced from digestion with A + B : {1, 1, 4, 4, 8, 9}.

Exercise 2. Try to determine the restriction map of a linear molecule that generates the following data:

- Products produced from digestion with A : {3, 5, 6, 6, 8};
- Products produced from digestion with B : {2, 3, 4, 4, 4, 4, 7};
- Products produced from digestion with A + B : {1, 1, 1, 2, 2, 2, 3, 3, 4, 4, 5}.

Write a description of what you did. Calculate the number of possible orderings. Don't test all possible orderings unless you are using a computer program!

Exercise 3. In Section 4.2.1, the products by digestion with A + B number $n + m - 1$. Show that this is correct. What is the number of A + B products if A and B have one cut site in common?

Exercise 4. Download the λ DNA sequences and determine to the base pair the exact length of the *EcoRI* restriction fragments. Note that since *EcoRI* is not a blunt end cutter you must describe your convention for measuring "length."

Exercise 5. Prove that n_{tot} , the total number of fragments produced by an incomplete digest of a linear DNA molecule, is $n_{\text{tot}} = \binom{n+1}{2}$, where n is the number of fragments produced in a complete digest. [Hint: The middle term can be obtained by writing down an index number for each site along left and top sides of a square and considering how many ways there are to choose pairs of sites that bracket an undigested segment.]

Exercise 6. Show that the method to solve DDP has time complexity $n! m! \times (n + m)$.

Exercise 7. Repeat the calculation done in Computational Example 4.1, $f_c = 0.95$, for the human genome, $G = 3.2 \times 10^9$, and for lambda inserts of length 20 kb.

Exercise 8. Show that the number of partial digestion fragments from a molecule that has n complete digestion fragments is $\binom{n+1}{2}$.

Exercise 9. The equation (4.6) for f_c is for a point locus. Find the probability that you would clone an entire locus of length l at least once in N random clones of length L (where $l < L$).

Exercise 10. What is c for 500,000 λ -clones of the human genome? How many λ -clones are required to represent 99% of the human genome?

Exercise 11. For complete digestion of a molecule with n restriction sites, there are $n + 1$ restriction fragments. For incomplete digestion, let $p = \mathbb{P}(\text{any site is cut})$. If all cuts of one molecule are independent, find $\mathbb{P}(\text{digestion of a molecule results in } k \text{ fragments}), 1 \leq k \leq n + 1$.

Exercise 12. Show that if $N \rightarrow \infty$, then $c \rightarrow \infty$, and evaluate $\lim_{c \rightarrow \infty} \Gamma = \lim N e^{-c(1-\theta)}$. Explain this result, given that complete coverage of the genome ($c \rightarrow \infty$) should produce one island. Can you correct formula (4.10)?

Genome Rearrangements

5.1 The Biological Problem

Genomes of living organisms consist of genes and noncoding DNA arranged in particular orders on one or more chromosomes. Eukaryotes usually have several pairs of linear chromosomes (humans have 23 pairs in each somatic cell), but prokaryotes typically have only one or a few circular (sometimes linear) chromosomes. The order of genes (or sets of genes) along a particular chromosome of a particular species is ordinarily conserved, and the list of genes or physical markers along a genome, together with the distances between these genes or markers, is called either a genetic or physical map, depending upon the units in which distances are measured. For each species, there is a particular genetic map that distinguishes it from other species. For example, even though the mouse (*Mus musculus*) contains most of the same types of genes found in humans, the genetic map of the mouse differs from that of humans. This is illustrated in Fig. 5.1 for human (*Homo sapiens*) chromosome Hsa6, which contains blocks of DNA sequences corresponding to portions of six different mouse chromosomes.

Clearly, segments of DNA in the two species were rearranged with respect to each other during their independent descent from a common ancestor. If blocks of sequences are excised from one location and inserted into another, those sequences are said to have been *translocated*. On the other hand, it is possible that the chunks of sequence in a syntenic block (Fig. 1.5) remain together but are *permuted* in one organism relative to the other. An example of permutation is blocks 1-2-3 of Hsa6 in Fig. 5.1, which appear in the order 3-1-2 in the mouse chromosome Mmu13. Sometimes sequences in the genome of an organism are duplicated elsewhere in the same genome (either on the same or on a different chromosome). For example, it is estimated that around 5% of the human genome (exclusive of transposable elements; see Chapter 14) consists of duplicated sequences (Bailey et al., 2002). Segmental and whole-genome duplications are discussed in Chapter 14.

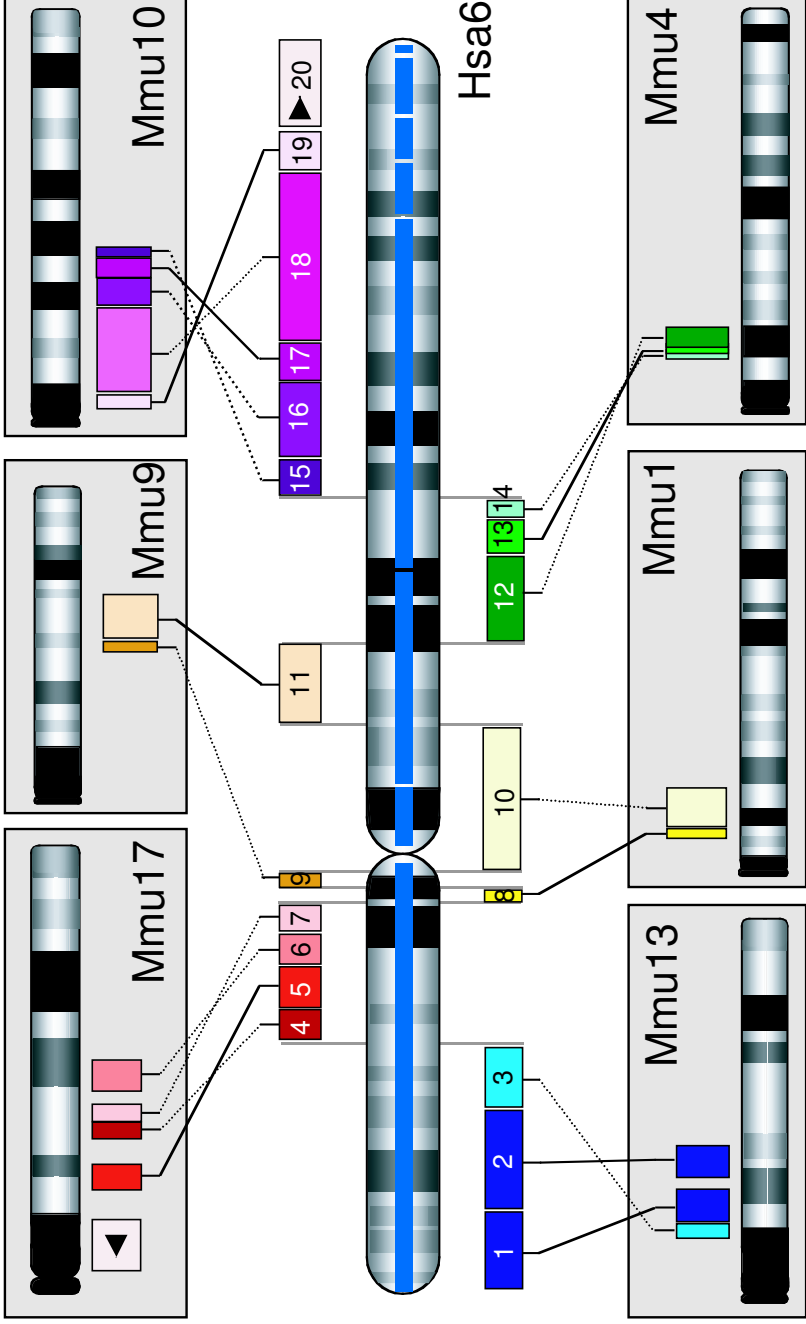
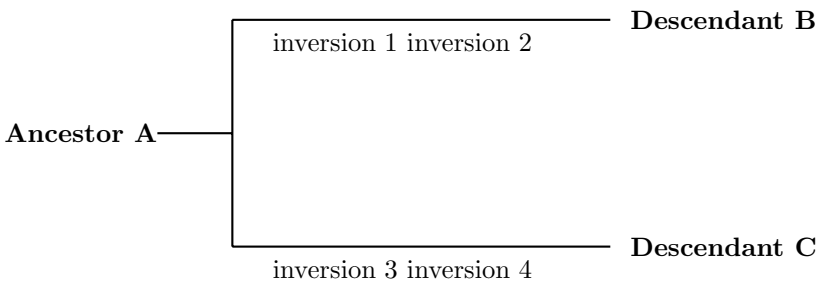


Fig. 5.1. [This figure also appears in the color insert.] Syntenic blocks conserved between human chromosome Hsa6 and mouse chromosomes. Broken lines indicate regions that appear in inverted orders in the two organisms. Reprinted, with permission, from Gregory SG et al. (2002) *Nature* 418:743–750. Copyright 2002 Nature Publishing Group.

5.1.1 Modeling Conserved Synteny

When comparing genomes of two organisms such as mice and humans, we sometimes observe local similarities in the gene content and gene order. If two or more genes that occur together on a single chromosome in organism B also occur together on a particular chromosome in organism C, this occurrence is an example of *conserved synteny* (see Fig 1.4 in Chapter 1). As shown in Fig. 5.1, the gene orders may or may not be conserved within syntenic blocks that are distributed among different chromosomes in a related organism. We will use a simple example to show how such patterns can arise by genome rearrangement.

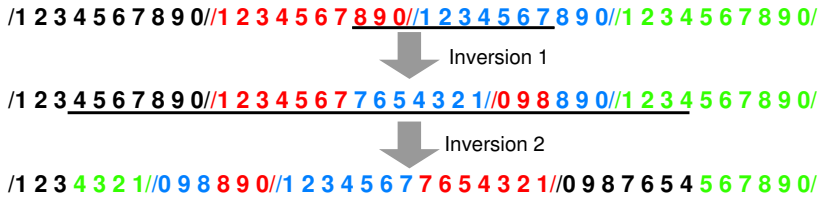
Figure 5.2 shows four different ancestral “chromosomes,” each distinguished by a different color. Each ancestral chromosome is of identical size, and each has ten “genes” 0, 1, . . . , 9. Each number/color combination represents a unique gene. This means that 2-green is not the same gene as 2-red. The telomeres are indicated by a single slash, /. We will model an evolutionary process indicated in the following diagram:



To model the genome rearrangements, we line up the A chromosomes in tandem and then invert different portions of this combined string in two independent steps to obtain chromosomes of Descendant B. The same process is repeated by inverting different segments to produce Descendant C. By inversion, we mean that we take a portion of the sequence of genes and reverse their order relative to the surrounding genes. For example, if the middle three genes represented by 1 2 3 4 5 were inverted, the result would be 1 4 3 2 5. Inversions of chromosomal regions are well-documented in most organisms for which there are genetic data.

Fig. 5.2 (Following page). [This figure also appears in the color insert.] Model producing conserved synteny in descendant chromosomes B and C after independent inversion steps starting with ancestral chromosome A. Telomeres (/) divide the “genes” (numbered) into four chromosomes. Chromosomes are placed end-to-end before inversions (reversals) are performed to simulate translocation. Two inversion steps on the path leading to B and two inversion steps on the path leading to C are sufficient to generate syntenic relationships resembling those seen in Fig. 5.1.

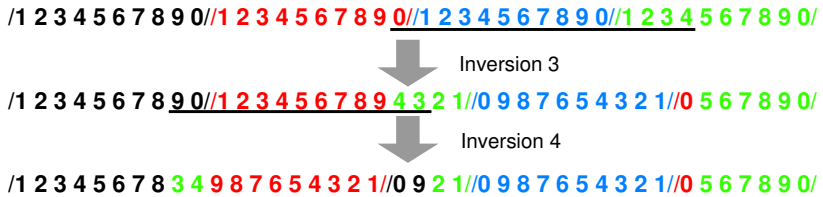
Ancestor A:



Chromosomes of Descendant B:

- B1 /1 2 3 4 5 6 7 7 6 5 4 3 2 1/
- B2 /0 9 8 7 6 5 4 5 6 7 8 9 0/
- B3 /1 2 3 4 3 2 1/
- B4 /0 9 8 8 9 0/

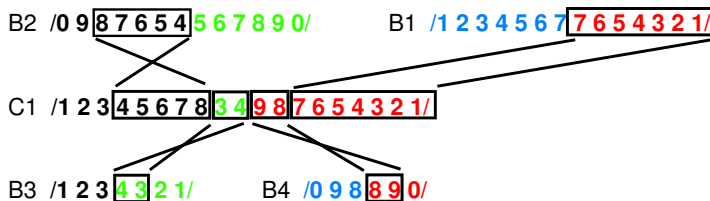
Ancestor A:



Chromosomes of Descendant C:

- C1 /1 2 3 4 5 6 7 8 3 4 9 8 7 6 5 4 3 2 1/
- C2 /0 9 8 7 6 5 4 3 2 1/
- C3 /0 5 6 7 8 9 0/
- C4 /0 9 2 1/

Relationships between chromosome C1 and chromosomes B1–B4:



Conserved synteny occurs when two or more genes occur together on any chromosome of Descendant B *and* on any chromosome of Descendant C. For example, 0-black and 9-black occur together on chromosome C4, and they also occur together on chromosome B2. 0-red and 5-green appear together on chromosome C3, but they do not appear together on any of the B chromosomes. This is *not* a conserved synteny. In the modelled example (Fig. 5.2, bottom), C1 shares regions of conserved synteny (and syntenic blocks) with B1, B2, B3, and B4.

Inversions of the type shown in this example can produce results like those illustrated in Fig. 5.1. The difference between Fig. 5.1 and our illustrative model is that in Fig. 5.1 each colored region represents many genes—not just one (i.e., 18-purple in Fig. 5.1 may contain hundreds of genes or more). Also Hsa6 has segments that are syntenic with six mouse chromosomes. The observed arrangements of genes on vertebrate chromosomes resemble what we would see if a process similar to the one modelled had occurred. In the example modeled, placing chromosomes end-to-end and then inverting segments corresponds to a translocation if the two breakpoints associated with the inversion fall within two chromosomes and to an inversion if they both fall within only one chromosome.

The model just considered should not be taken as the actual mechanism for chromosome evolution. It merely illustrates how one type of rearrangement process can produce results similar to what is observed in contemporary eukaryotic chromosomes. However, rearrangements like these have been observed in closely related contemporary species. For example, human chromosomes 1 and 18 have large inversions compared with the corresponding chimpanzee chromosomes, and chimpanzee chromosomes 12 and 13 together contain material that is located in human chromosome 2 (see Olson and Varki, 2003, for a review comparing chimpanzees and humans). Chromosome rearrangements thus may be used to help track evolutionary processes.

5.1.2 Rearrangements of Circular Genomes

As we mentioned above, bacteria often have circular chromosomes. Circular chromosomes are also found in mitochondria and chloroplasts. Remember that mitochondria and chloroplasts are organelles that are thought to have been reduced from eubacterial endosymbionts. These organelles contain many copies of their respective genomes. For example, human mitochondria contain approximately 10^3 - 10^4 DNA circles, each about 16,000 bp long and containing 37 genes. Circular chromosomes or organelle DNAs of related organisms may not have identical gene arrangements. Because of their circular structures, it is possible to invert segments of circular chromosomes by a single crossover event, which produces an **inversion**, a permutation of the original sequence with two novel junctions, or breakpoints. When this occurs repeatedly, considerable “shuffling” of the gene orders may occur.

One of the most interesting applications of computational biology is the reconstruction of phylogenies of organisms (see Chapter 12). This is often accomplished by using the actual DNA sequences of genes or gene segments. Human mtDNA sequences have been used to reconstruct the ancestor-descendant relationships leading to contemporary human populations (see, for example, Fig. 12.2). Organellar DNAs are particularly useful sources of ancient DNA samples because the copy numbers of mtDNA sequences are thousands of times more abundant than particular sequences from nuclear DNA. But one problem with organellar DNA and other small circular DNAs is that they may have high mutation rates, which makes it hard to establish with confidence the time separating distantly-related gene homologs. One way of dealing with this problem is to delineate blocks of conserved genes, or *conserved segments*. Evolutionary relationships are then analyzed in terms of these conserved segments rather than the actual DNA sequence. In this chapter, we will show how genome sequences can evolve by inversion (which we will be calling these **reversals**), and we will be estimating the **distance** between two genomes by using the number of reversals required to convert the order of conserved segments in one genome into the order found in the other. A set of distances relating a collection of genomes can be used to construct phylogenetic trees, which are graphs of ancestor-descendant relationships between these genomes. These are discussed in detail in Chapter 12.

5.2 Permutations

5.2.1 Basic Concepts

To begin, we analyze rearrangements of linear, unichromosomal genomes (viral, organellar, bacterial), with each segment representing a group of contiguous genes (a conserved segment). As noted, many of these genomes are circular but have an analog linear genome, as otherwise the analysis is needlessly complicated. Because each conserved segment contains several genes, we can define an orientation for each segment. Before we deal with orientation, we first discuss the analysis of ordered elements, g_1, g_2, \dots, g_n . In Section 5.3, each of these elements g_i will be identified with conserved segments, and the permutation G will be identified with a particular genome.

The identity permutation of these elements is defined as the permutation in which the indices increase in numerical sequence:

$$I : g_1 g_2 g_3 \dots g_n.$$

The elements may also appear as a permutation of the elements in the identity permutation. For example, if $n = 7$, permutation G might have elements in the following order:

$$G : g_3 g_7 g_2 g_6 g_4 g_1 g_5.$$

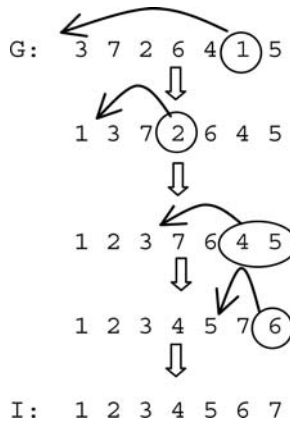
For simplicity, we represent this particular permutation by just listing the index numbers,

$$G: \quad 3 \ 7 \ 2 \ 6 \ 4 \ 1 \ 5,$$

and we represent the identity permutation similarly:

$$I: \quad 1 \ 2 \ 3 \ 4 \ 5 \ 6 \ 7.$$

How can we transform G into I ? One way is to pick up the numbers one or more at a time and drop them into the correct position. The equivalent biological process is translocation. Transformation of G to I by translocation is diagrammed as:



As indicated above, a common way for DNA molecules to rearrange is by inversion. The equivalent term in computer science is *reversal*. An example of a reversal performed on I is illustrated below, where the interval 3 4 5 is reversed:

$$1 \ 2 \ \boxed{5 \ 4 \ 3} \ 6 \ 7$$

The order of the elements in the box is reversed from the order in which they were found in the identity permutation. In general, if we are given a permutation:

$$g_1 \ g_2 \ g_3 \ \dots \ g_{i-1} \ g_i \ g_{i+1} \ \dots \ g_{j-1} \ g_j \ g_{j+1} \ \dots \ g_n,$$

a reversal of the interval $[i, j]$ yields the permutation

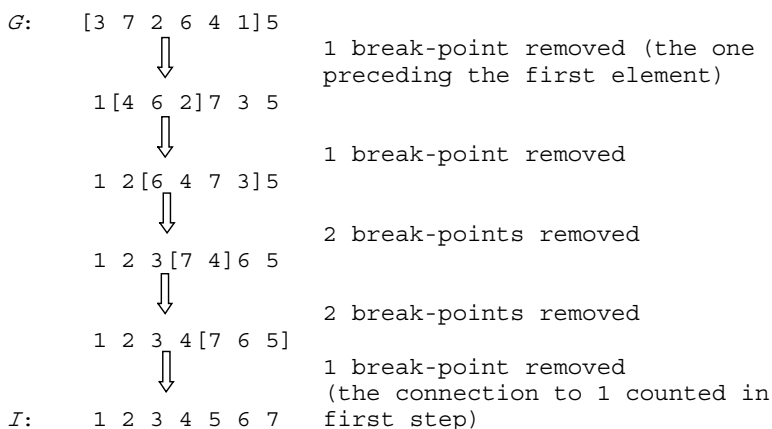
$$g_1 \ g_2 \ g_3 \ \dots \ g_{i-1} \ [g_j \ g_{j-1} \ \dots \ g_{i+1} \ g_i] \ g_{j+1} \ \dots \ g_n.$$

The order of the elements in brackets has been reversed. Adjoining elements may represent either an adjacency or a breakpoint.

$$\begin{aligned}
 g_k g_j &\Rightarrow \textit{Adjacency} \text{ if } j = k \pm 1, \\
 g_k g_j &\Rightarrow \textit{Breakpoint} \text{ if } j \neq k \pm 1.
 \end{aligned}$$

Since in the second string immediately above g_{i-1} is followed by g_j , there is one breakpoint after g_{i-1} . Similarly, since g_i is followed by g_{j+1} , there is another breakpoint just before g_{j+1} . The inversion has produced two breakpoints.

Let's now return to our previous example $G = 3\ 7\ 2\ 6\ 4\ 1\ 5$. We want this to represent a circular array of permuted elements. This means that the first element in our linear representation is connected to the last element. If the first element is in the correct position, then we do not consider it to be preceded by a breakpoint, even if the last element is not yet in the correct position. (We don't want to count the same breakpoint twice.) By our definition of breakpoint shown above, you can see that there are seven breakpoints. Instead of transforming G to I by transposition of elements, we can effect this transformation by successive inversions (reversals) of two or more elements (brackets enclosing the elements to be inverted in each step):



As we see from this example, we have removed the seven breakpoints that were in the original sequence. It should be clear that the maximum number of breakpoints that can be removed by a reversal is 2. Note that the reversals do not always remove two breakpoints.

We define $d(G, I)$ as the *minimum number of reversals* required to convert G into I . Notice that reversals destroy breakpoints during the transformation process. If the *number of breakpoints* in G is $b(G)$, then it should be clear that

$$d(G, I) \geq b(G)/2.$$

Also, each reversal can remove at least one breakpoint, so

$$d(G, I) \leq b(G)$$

and it follows that

$$b(G)/2 \leq d(G, I) \leq b(G).$$

Let's check this against the example that we just used. We should recognize that there is a breakpoint just before 3 in G (that element should have been

1 if there had been no breakpoint). Using the same reasoning, there is also a breakpoint after the last element. In fact, $b(G) = 8$, and $d(G, I) \geq 8/2$. We know that it will take at least four reversals to transform G to I . In the illustration, we employed five reversals.

What is the greatest number of reversals required to transform the most unfavorable permutation G to I (worst-case scenario)? What is the least number of reversals it would take to transform G to I ? We can easily estimate what the greatest number of reversals (worst case) would be. We can imagine that you start from this worst-case permutation by reversing the set of elements from the initial one in the permutation up to and including g_1 . This puts g_1 in front. Then reverse all elements beginning with the second in the permutation up to and including g_2 . Continue the process until the identity permutation is achieved. In the worst case, we will have to do this for $n - 1$ elements. We don't have to do a separate operation for g_n because if the first $n - 1$ elements are sorted as described, then n automatically falls into the last position. It can be shown that, for some permutations, the *least* number of reversals required is $(n + 1)/2$. Thus we have

$$(n + 1)/2 \leq d(G, I) \leq n - 1.$$

Examples of the worst case are

$$G_{13} = 3\ 1\ 5\ 2\ 7\ 4\ 9\ 6\ 11\ 8\ 12\ 10\ 13$$

and

$$G_{14} = 3\ 1\ 5\ 2\ 7\ 4\ 9\ 6\ 11\ 8\ 13\ 10\ 12\ 14.$$

5.2.2 Estimating Reversal Distances by Cycle Decomposition

Given X and Y , which are permutations of the same letters, how can we determine $d(X, Y)$? We describe a graphical method for doing this, and as an example (to keep the graphs simple) we estimate the reversal distance between F and I , where

$$F : \quad 1\ 2\ 4\ 5\ 3 \qquad I : \quad 1\ 2\ 3\ 4\ 5.$$

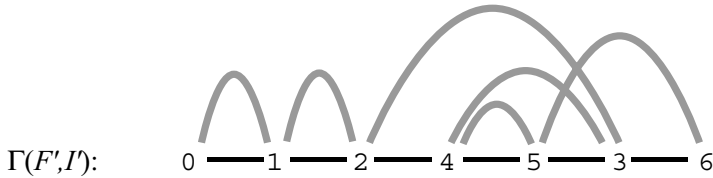
First, extend each permutation by adding 0 before the first element and $n + 1$ after the last element, where n is the number of elements. These changes do not alter the reversal distances between the permutations because the elements are in their correct positions.

$$F' : \quad 0\ 1\ 2\ 4\ 5\ 3\ 6 \qquad I' : \quad 0\ 1\ 2\ 3\ 4\ 5\ 6.$$

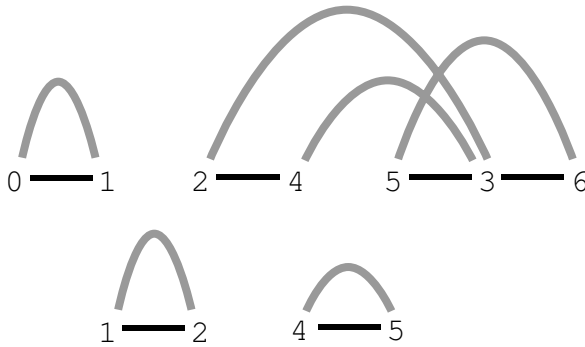
Each element index corresponds to a **vertex** of the graph, and lines connecting the vertices are called **edges**. First, we connect all adjacent vertices of F' with black edges:

$$F' : \quad 0 \text{ — } 1 \text{ — } 2 \text{ — } 4 \text{ — } 5 \text{ — } 3 \text{ — } 6$$

Now, in the same way, connect all of the adjacent I' vertices with grey edges.



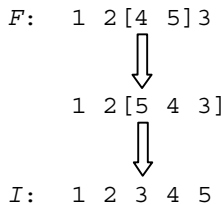
The diagram above is called a graph, $\Gamma(F', I')$. This type of graph is said to be **balanced** because the number of black edges at each vertex is equal to the number of grey edges. It can be decomposed into **cycles** that have edges with alternating colors and that don't share edges with other cycles (i.e., they are **edge-disjoint**). We do not give an algorithm for this decomposition, which in general is a challenging problem. We define $c(F, I)$ as the maximum number of alternating, edge-disjoint cycles into which $\Gamma(F, I)$ can be decomposed. There are four in this case, and they are shown in the following diagram:



The reason that we did this is that for most of the usual and interesting biological systems, the equality holds for the next expression for reversal distance between two permutations X and Y :

$$d(X, Y) \geq n + 1 - c(X, Y) = \tilde{d}(X, Y).$$

The general problem, just like the TSP, is NP-complete. In the case of permutation F , the number of elements n was equal to 5, and we found that $c(F, I) = 4$. This means that $\tilde{d}(F, I) = 2$. In other words, we should be able to transform F into the identity permutation with just two reversals. They are shown below (again, we reverse the order of elements between the brackets):

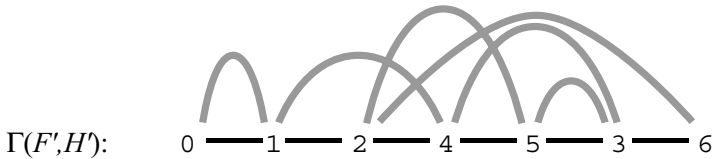


5.2.3 Estimating Reversal Distances Between Two Permutations

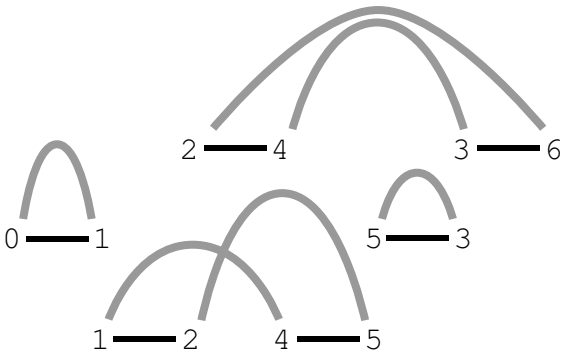
Up to now, we have been determining the reversal distance $d(G, I)$ between permutation G and the identity permutation I . Now suppose that we have the two permutations F and H , with the following elements:

$$F : 1 \ 2 \ 4 \ 5 \ 3 \qquad H : 1 \ 4 \ 3 \ 5 \ 2.$$

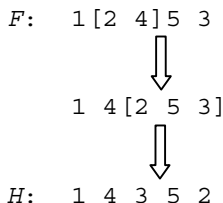
The distance between F and H , $d(F, H)$ and $\tilde{d}(F, H)$, is calculated just as was done above, except that this time the second set of edges will not be drawn connecting vertices in increasing numerical order but instead *in the order in which these vertices appear in H* . For simplicity, we again represent each element by its numerical index and, as before, we extend each permutation by prefixing it with 0 and by adding $n + 1$ at the end (6 in this case). We connect the vertices of F with black edges, taking vertices in succession as they appear in F . Then we add grey edges between vertices, *taking them in the order in which they appear in H* . The result is:



The decomposition of this graph into the maximum number of edge-disjoint cycles is:



Since $\Gamma(F', H')$ can be decomposed into a maximum of four cycles, $\tilde{d}(F, H) = 5 + 1 - 4 = 2$, and we expect that F can be transformed into H with two reversals. This is shown in the following diagram:



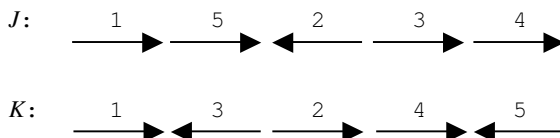
5.3 Analyzing Genomes with Reversals of Oriented Conserved Segments

In the previous section, we showed how to analyze permutations. When we start to analyze genomes, the elements are now genes (or sets of genes in an interval that contains no breakpoints), and these have *orientation*. Gene orientation might be indicated by transcription direction, which might be denoted as “+” if it is transcribed left to right in a particular map as drawn and “-” if it is transcribed right to left. Genes α , β , and γ might appear in one genome in an order $+\alpha + \beta + \gamma$ but in another genome in the reverse order $-\gamma - \beta - \alpha$. These might represent conserved segment g_i , which would be represented as $+g_i = +\alpha + \beta + \gamma$ in the former case and as $-g_i = -\gamma - \beta - \alpha$ in the latter case.

For example, consider the two genomes J and K below:

$$J: +g_1 +g_5 -g_2 +g_3 +g_4 \qquad K: +g_1 -g_3 +g_2 +g_4 -g_5.$$

These could be represented graphically as (again, employing only the subscript labels):

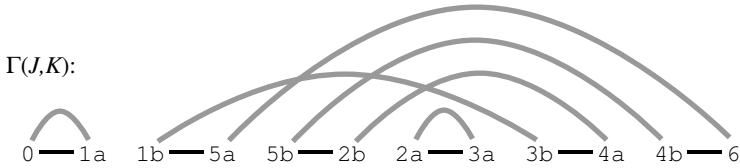


As before, we are seeking the reversal distance $d(J, K)$, which is the minimum number of reversals needed to convert J to K . But now when we do reversals to sort the segment orders, we must also keep track of the orientations of the conserved segments to make sure that both gene order *and* orientation are correct. Just as we did before, we calculate $d(J, K)$ by constructing an edge-colored graph and then counting the maximum number of alternating cycles into which the graph can be decomposed. But we must modify the procedure to keep track of the orientations of the conserved segments. (As noted above, the genome rearrangement problem with unsigned segments is NP-complete. Although signed segments would appear to make the problem more difficult, actually there is an $O(n^2)$ algorithm! Further discussion of this is beyond the scope of this chapter.)

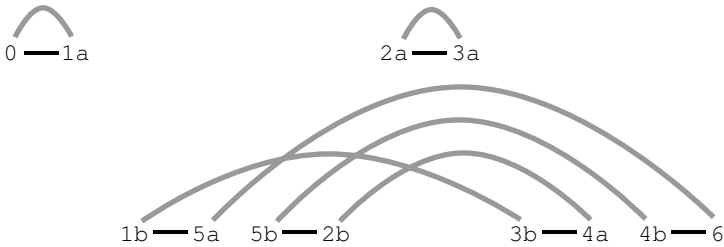
To track the orientation of conserved segment i , label its left end (tail) ia and its right end (head) ib . If genome J has n conserved segments, there are now $2n$ elements representing their ends, thus specifying their location and orientation. We prefix the list of elements in J with 0 and add element $n+1$ to the right end as before to yield a total of $2n+2$ elements. Genomes J and K are each represented by a particular permutation of the same $2n+2$ elements, and the approaches described in the previous section can now be applied. Our example above now becomes

$$\begin{aligned}
 J: & \quad 0 \quad 1a \quad 1b \quad 5a \quad 5b \quad 2b \quad 2a \quad 3a \quad 3b \quad 4a \quad 4b \quad 6, \\
 K: & \quad 0 \quad 1a \quad 1b \quad 3b \quad 3a \quad 2a \quad 2b \quad 4a \quad 4b \quad 5b \quad 5a \quad 6.
 \end{aligned}$$

Now we create an edge-colored graph, but we only draw edges where adjacent conserved segments are connected (i.e., we won't draw edges between the ends of the same conserved segment, ia and ib). We designate edges between conserved segments in J with black lines and edges between conserved segments in K with grey lines.



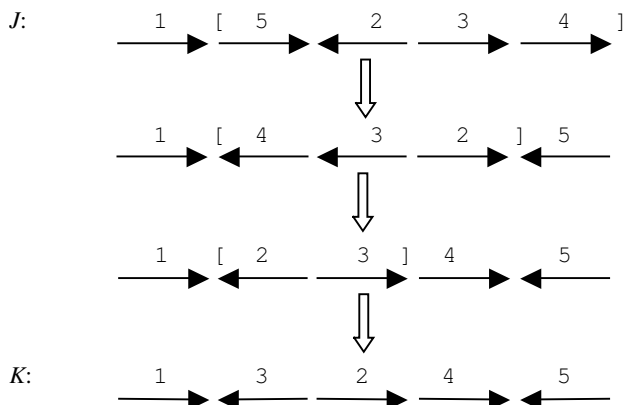
The decomposition of this graph into edge-disjoint alternating cycles is:



We find that there are three such cycles, and thus we calculate the reversal distance between J and K :

$$\tilde{d}(J, K) = n + 1 - c(J, K) = 5 + 1 - 3 = 3.$$

We then expect to be able to transform J to K by using three reversals. These are shown in the following diagram (with $[]$ indicating set of conserved segments being reversed):



We notice that the alternating color edge-disjoint decomposition into the maximum number of cycles is now easy. This is a general property of the analysis of signed reversals. It is accomplished by choosing an edge and following the alternating edge-vertex path until returning to the beginning. There are no choices to make. A challenging problem became easy with the addition of biological detail.

Let's remember again at this point why we are going through all of this. What we are extracting is the set of $\bar{d}(X, Y)$ from a collection of related genomes. These are pairwise differences, or distances. It turns out that if we have a collection of m genomes and all of the $m(m-1)/2$ pairwise differences between them, we can build a phylogenetic tree that reveals the relationships among the genomes. This sort of approach has been used by different investigators to analyze mitochondrial genomes (Sankoff et al., 1992) and herpesvirus genomes (Hannenhalli et al., 1995). A discussion of approaches to genome reconstruction is provided by Bourque and Pevzner (2002).

Computational Example 5.1: Determination of reversal distance $d(X, Y)$ between genome X and genome Y

Given two related genomes containing the same set of conserved segments, but in different orders and orientations:

- Step 1: Label the ends of each conserved segment i as ia and ib , for all $1 \leq i \leq n$, where n is the number of conserved segments. These labeled ends will be the vertices of the graph.
- Step 2: Write down in order the ends of the oriented conserved segments in the sequence that they occur in genome X .
- Step 3: Add 0 at the beginning of the permutation representing genome X , and add $n+1$ at the end of the permutation.
- Step 4: Repeat steps 2 and 3 for genome Y .

- Step 5: Connect vertices from *different* fragments with a black line if they are joined in genome X . Connect vertices from *different* fragments with a grey line if they are joined in genome Y . (No connections will be made for ia and ib .)
- Step 6: Decompose the graph generated in step 5 into the maximum number of edge-disjoint alternating cycles. This number is $c(X, Y)$.
- Step 7: Compute the reversal distance, the minimum number of reversals of one or more conserved segments needed to transform one genome into the other: $\tilde{d}(X, Y) = n + 1 - c(X, Y)$.

5.4 Applications to Complex Genomes

The illustrations above applied to unichromosomal linear genomes, which can be generalized to circular genomes of eukaryotic organelles and many bacteria. Relating multichromosomal organisms by genome rearrangements is much more complicated, although this has been done for humans and mice. We will illustrate these more complex problems without going into detail. Genome rearrangements are also discussed in Chapter 14. We continue to use the mouse and human genomes as examples (MGSC, 2002; Pevzner and Tesler, 2003).

5.4.1 Synteny Blocks

In the first and second sections of this chapter, we discussed conserved synteny and conserved segments (see also Fig. 1.5 and Fig. 14.8). We need to be a bit more precise to accommodate the data. According to one definition (see Section 14.3 for other definitions), a **conserved segment** is a region found in two genomes in which homologous genes retain the same order and relative map position in both genomes. Does it follow that conserved segments contain *no* gene rearrangements? It is estimated that there are about 25,000 mammalian genes in genomes of about 3×10^9 bp. This means that, on average, conserved segments that include three genes would exceed 300,000 bp. Is it possible that there have been rearrangements in intervals of size 200,000 to 1,000,000 bp that would not be detected in conserved segments?

The availability of sequenced genomes makes possible more precise measures of conserved synteny and conserved segments, based upon a higher density of physical markers. Regions of local similarity between two sequenced genomes can be readily identified (sequence similarity and alignment are discussed in the next chapter). Suppose that subsequence f_i in genome F is similar to subsequence g_i in genome G , and that (a) alignment of f_i to G yields g_i as the highest-scoring alignment and that (b) alignment of g_i to F yields f_i as the highest-scoring alignment (i.e., f_i and g_i are reciprocal “best hits”). The subsequences f_i and g_i can be used as **sequence anchors** or

landmarks to aid in comparing the two genomes. In principle, there can be many more sequence anchors than there are genes and, unlike genes, their identification is uninfluenced by errors in gene annotation (e.g., incorrectly identifying a gene in F as an ortholog of a gene in G —see Chapter 14 for how orthologs are identified). In a recent study relating the mouse and human genome organizations, nearly 560,000 genome anchors were employed.

Whereas conserved segments were defined above in terms of genes, synteny blocks can also be defined in terms of anchors. A **syntenic block** is a set of sequence anchors that appear together (but not necessarily in the same order) in the genomes from two different organisms. Because there are so many more anchors than there are genes, the synteny blocks are based upon much finer partitioning of the chromosome than are conserved segments defined by genes. Conserved segments may contain short inversions or other types of *microrearrangements* that are not detected by mapping with sequence anchors. Synteny blocks can be used to define *macrorearrangements* relating two chromosomes by approaches similar to those we have discussed above (Pevzner and Tesler, 2003).

5.4.2 Representing Genome Rearrangements

The methods we have discussed for single chromosomes have relied on making what were, in effect, one-dimensional diagrams. When comparing two chromosomes or genomes, it is possible to represent the relationships by using two-dimensional diagrams, with each axis representing the synteny blocks for each organism. The projection of the two-dimensional graph onto either axis produces a unidimensional diagram of the type that we have employed up to now. Figure 5.3 shows an example of a two-dimensional diagram comparing the human and mouse X chromosomes. The problem to be solved is exactly the same as the one that we have discussed in the unichromosomal case: determining the distance between the chromosomes by enumerating the number of reversals required to convert the order of synteny blocks found in mice into the order found in humans. This is of course just a part of the larger problem, identifying the total rearrangement distance between the complete genomes. Exercise 9 illustrates cycle decomposition in two dimensions.

To demonstrate how the observations recorded in Fig. 5.3 could have arisen, we perform a simple simulation.

Computational Example 5.2: Simulating genome rearrangements

Using R, provide an ordered set of 100 anchor loci for genome x . This might represent the ancestral genome. We will not perform any rearrangements on x .

```
x<-c(1:100)
```

Next we provide an ordered set of 100 anchor loci for genome y :

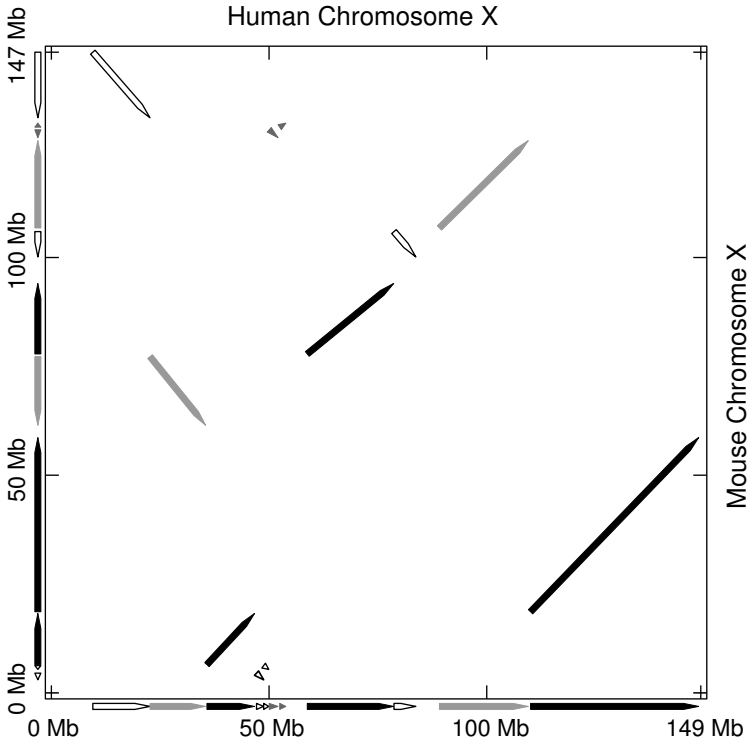


Fig. 5.3. Synteny blocks shared by human and mouse X chromosomes. The arrow-head for each block indicates the direction of increasing coordinate values for the human X chromosome. Reprinted, with permission, from Pevzner P and Tesler G (2003) *Genome Research* 13:37–45. Copyright 2003 Cold Spring Harbor Laboratory Press.

```
y<-c(1:100)
```

We now transform genome y into genome y_4 through a series of four reversal steps of our choice, saving each intermediate genome y_1 , y_2 , and y_3 . y_2 is produced by applying a second reversal to y_1 , which resulted from the first reversal, and so on.

```
y1<-y
y1[11:75]<-y1[75:11]
#Reversal of interval 11 to 75 inclusive in y

y2<-y1
y2[15:30]<-y2[30:15]
#Reversal of interval 15 to 30 inclusive in y1
```



```

y3<-y2
y3[55:80]<-y3[80:55]
#Reversal of interval 55 to 80 inclusive in y2

y4<-y3
y4[70:90]<-y4[90:70]
#Reversal of interval 70 to 90 inclusive in y3

```

Now we plot the results on a single plot with three rows of two columns (`mfrow=c(3,2)`) and using “.” as the print character for each point (`pch="."`).

```

par(pin=c(2.5,2.5), mfrow=c(3,2),pch=".")
plot(x,y, main="A.")
plot(x,y1, main="B.")
plot(x,y2, main="C.")
plot(x,y3, main="D.")
plot(x,y4, main="E.")

```

The results of this simulation are plotted in Fig. 5.4. We can see step by step how a graph such as the one shown in Fig. 5.3 can be generated by reversals.

We note in passing that these graphs are analogous to the “dot plots” that will be discussed in Chapter 7. In that chapter, the plots are of k -word matches in two different sequence strings. Here, the plots are of synteny blocks in two different genomes. Interpretations in terms of diagonals are very similar in both cases.

5.4.3 Results from Comparison of Human and Mouse Genomes

Mouse and human genomes have been compared by Pevzner and Tesler (2003), who also present an extremely complicated multichromosomal breakpoint graph. They identified 281 synteny blocks (within stated gap and size criteria), and they concluded that the order of the mouse genome synteny blocks can be converted into the order seen in the human genome by 245 genome rearrangements: 149 inversions, 93 translocations, and 3 fissions. Mice and humans share a common ancestor estimated to have lived about 83 million years ago. Since that time, there have been about 1.5 rearrangements occurring per million years of evolutionary time separating the two species (245 rearrangements \div (2×83 million years)); the factor of 2 reflects the fact that evolution is occurring in both lineages).

As more mammalian genomes are sequenced, comparisons of their sequence organizations will provide additional measures of genetic distance, which can be used together with DNA sequence differences to delineate evo-

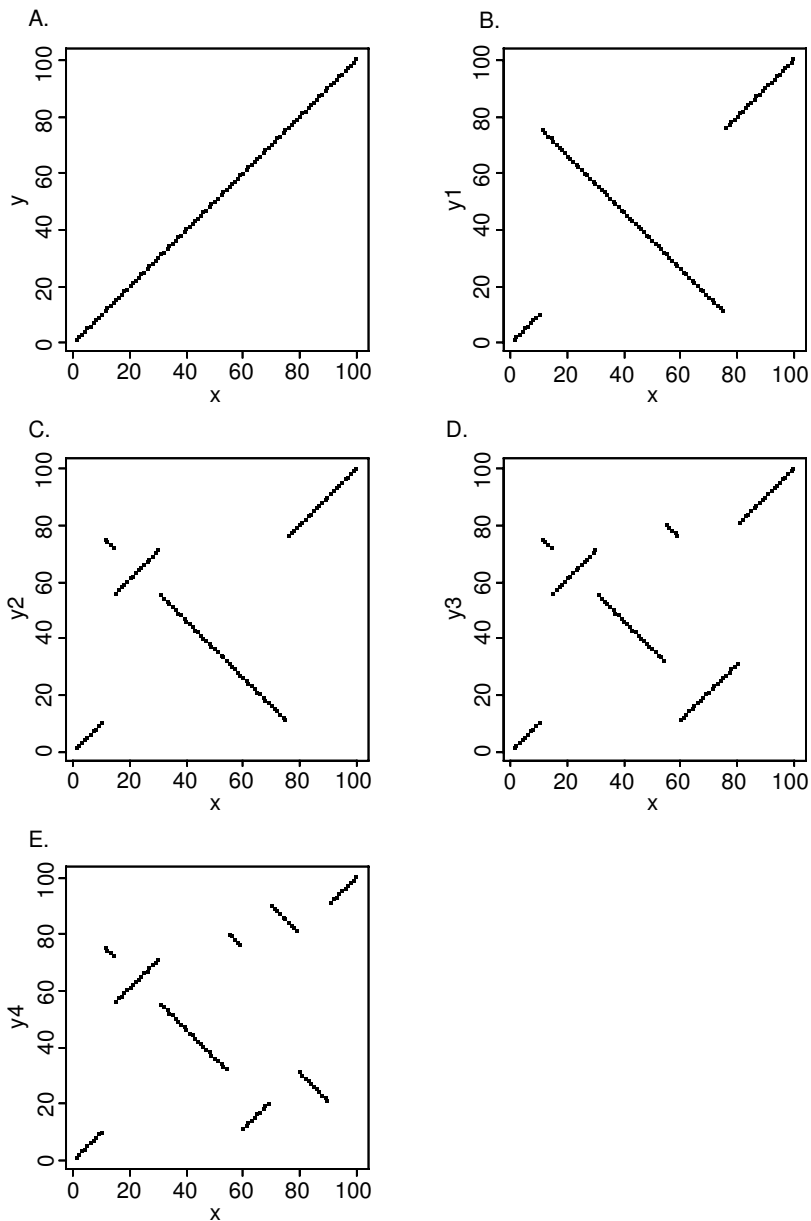


Fig. 5.4. Simulation of rearrangement of a single chromosome by reversals. Anchor loci are plotted along axes. Panel A represents chromosomes x and y in the unarranged states. Chromosomes y1, y2, y3, and y4 in panels B through E are related to chromosome y by one reversal relative to the previous genome (e.g., y1 has one reversal compared to y, y2 has one reversal compared to y1, etc.).

lutionary relationships. General approaches for genome comparisons are described in Chapter 14.

References

- Bailey JA et al. (2002) Recent segmental duplications in the human genome. *Science* 297:1003–1007.
- Bourque G, Pevzner PA (2002) Genome-scale evolution: Reconstructing gene orders in the ancestral species. *Genome Research* 12:26–36.
- Gregory SG et al. (2002) A physical map of the mouse genome. *Nature* 418:743–750.
- Hannenhalli S, Chappey C, Koonin EV, Pevzner PA (1995) Genome sequence comparison and scenarios for gene rearrangements: A test case. *Genomics* 30:299–311.
- Mouse Genome Sequencing Consortium (2002) Initial sequencing and comparative analysis of the mouse genome. *Nature* 420:520–562.
- Olson MV, Varki A (2003) Sequencing the chimpanzee genome: Insights into human evolution and disease. *Nature Reviews Genetics* 4:20–28.
- Pevzner P, Tesler G (2003) Genome rearrangements in mammalian evolution: Lessons from human and mouse genomes. *Genome Research* 13:37–45.
- Sankoff D, Leduc G, Antoine N, Paquin B, Lang BF, Cedergren R (1992) Gene order comparisons for phylogenetic inference: evolution of the mitochondrial genome. *Proceedings of the National Academy of Sciences USA* 89:6575–6579.

Exercises

Exercise 1. For $X = 1\ 2\ 3\ 4\ 5\ 6\ 7$ and $Y = 2\ 3\ 1\ 4\ 6\ 5\ 7$, (a) find the break-points, (b) find the breakpoint graph, and (c) perform the cycle decomposition of the breakpoint graph.

Exercise 2. Taking $x = 1\ 2\ 3\ 4\ 5\ 6$ as the “reference genome” find the breakpoint graph for $y = 3\ 1\ 5\ 2\ 6\ 4$. What is $\tilde{d}(X, Y)$?

Exercise 3. Taking $X = +1\ +2\ +3\ +4\ +5\ +6$ as the reference genome, find the breakpoint graph for $y = +3\ -1\ +5\ +2\ -6\ +4$

Exercise 4. Find a set of reversals for the “ $n - 1$ ” distance for G_{13} (Section 5.2). Do the same thing for G_{14} .

Exercise 5. Find the cycle decomposition of $G_7 = 3\ 1\ 5\ 2\ 7\ 4\ 6$, for which the reversal distance is $7 - 1 = 6$.

Exercise 6. Given $G : 2\ 4\ 5\ 3\ 1$, a set of reversals that transform G to the identity permutation I is

$$\begin{array}{c}
 G: [2\ 4\ 5\ 3\ 1] \\
 \Downarrow \\
 1\ [3\ 5\ 4\ 2] \\
 \Downarrow \\
 1\ 2\ [4\ 5\ 3] \\
 \Downarrow \\
 1\ 2\ 3\ [5\ 4] \\
 \Downarrow \\
 I: 1\ 2\ 3\ 4\ 5
 \end{array}$$

Show by cycle decomposition that this set of reversals does not correspond to the minimum $d(G, I)$.

Exercise 7. The two-dimensional plot shown in Fig. 5.4D can be expressed as a one-dimensional graph using the conventions in Section 5.3. Recall that in the simulation that led to Fig. 5.4D, sequence x was not rearranged.

- Label the ends of the i th line segment ia and ib using the order and orientation of segments in x to determine the values of i and the assignments a or b .
- Make a diagram showing y_3 as a set of oriented arrows with the appropriate labelling. (For the purposes of this problem, the arrows can all be of the same length. See Section 5.3.)
- Draw a graph corresponding to the oriented sequence blocks in part b of this problem, and perform a cycle decomposition to verify that the computed reversal distance agrees with the actual number of reversals used to produce Fig. 5.4D (Computational Example 5.2).

Exercise 8. Reproduce Fig. 5.4D, except this time allow enough space to add a point labeled 0 in the lower left-hand corner and a point labeled 8 in the upper right-hand corner, with a gap separating these points from the first and last oriented segments, respectively. Shorten segments 1–7, and add the labeling as in Exercise 7a.

- Project all ia and ib onto the y_3 axis. (Each point should be separated from its neighbors by a space because the segments were shortened above.) Connect adjacent points with broken lines, except for points belonging to the same line segment.
- Using solid line segments, connect the ends of segments 1–7 in the order of appearance in x (i.e., connect ib to $(i + 1)a$, etc.), and project these solid line segments as arcs onto the y_3 axis.

- c. Compare the results of parts a and b of this problem with the graph that you drew in Exercise 7c. Are they the same (except for the lengths of edges connecting the points)?

Exercise 9. This exercise shows how to perform cycle decomposition in two dimensions, again using Fig. 5.4D. Redraw Fig. 5.4D just as you did in Exercise 8, with points 0 and 8 added, and ia and ib labels.

- a. In the two-dimensional area, use solid lines to connect adjacent segments in the order that they occur in x . (Connect 0 to $1a$ and $7b$ to 8 with arcs that are concave downward.)
- b. Use broken lines to connect adjacent segments in the order that they occur in $y3$. (Connect 0 to $1a$ and $7b$ to 8 with arcs that are concave upward)
- c. Now remove all segments 1–7 (corresponding to our not connecting the ia and ib vertices when i is the same). Decompose the composite set of broken and solid lines into disjoint cycles. (Cycles will contain both solid and broken lines.) Use the number of cycles to compute $d(x, y3)$. Does this agree with the result of Exercise 7c and the number of reversals used to produce $y3$ in the simulation?

Exercise 10. Use the approach of Exercise 9 to compute the distance separating the human and mouse X chromosomes (Fig. 5.3). (To prevent confusion, make all segments the same length, so that their projections on the coordinate axes would be separated by gaps.)

Exercise 11. GRIMM is the genome rearrangements Web server that computes optimal rearrangement scenarios relating a source and destination genome. It is available at <http://www-cse.ucsd.edu/groups/bioinformatics/GRIMM>.

- a. Represent genomes x and $y4$ in Fig. 5.4E as oriented arrows (see Exercise 7). Input x as the source genome and $y4$ as the destination genome into GRIMM and run the application. Do you get the same rearrangement scenario as was actually employed to generate Fig. 5.4E? Do you obtain the same number of reversals?
- b. Check your answer to Exercise 10 by applying GRIMM to the data in Fig. 5.3.

Sequence Alignment

6.1 The Biological Problem

Much of biology is based on recognition of shared characters among organisms, extending from shared biochemical pathways among eukaryotes to shared skeletal structures among tetrapods. The advent of protein and nucleic acid sequencing in molecular biology made possible comparison of organisms in terms of their DNA or the proteins that DNA encodes. These comparisons are important for a number of reasons. First, they can be used to establish evolutionary relationships among organisms using methods analogous to those employed for anatomical characters. Second, comparison may allow identification of functionally conserved sequences (e.g., DNA sequences controlling gene expression). Finally, such comparisons between humans and other species may identify corresponding genes in model organisms, which can be genetically manipulated to develop models for human diseases.

Genes or characters in organisms B and C that have evolved from the same ancestral gene or character in A are said to be **homologs**. Similar characters that result from independent processes (i.e., convergent evolution) are instances of *homoplasy*. Examples are the dorsal fins of sharks and whales. We are most interested in homologies because they usually exhibit conserved functions. Since organisms have greater or lesser degrees of evolutionary relationships, we anticipate that homologs will be found for most genes in organisms that are evolutionarily close. Thus, we might expect that mice would have homologs of human genes for immunoglobulins but would not expect such genes to occur in bacteria.

It is important to distinguish between sequence *homology* and sequence *similarity*. Suppose that genes g_B and g_C are homologs derived from gene g_A in organism A. They will have independently accumulated mutations along the paths $A \rightarrow C$ and $A \rightarrow B$. If the divergence between these two lineages was relatively recent, then the coding sequences for g_B and g_C will display the same character at most positions in their corresponding DNA or protein sequences. If the divergence between lineages was more remote in time, there will be a

lesser correspondence in characters at each position. **Similarity** refers to the degree of match at corresponding positions in the two sequences. This is often expressed as a percentage. Similarity is an expected consequence of homology, but when comparing two short sequences, it is possible for similarity to occur by chance. Similarity does not necessarily imply homology.

Identification of homologs makes it possible to trace evolutionary patterns of genes, which may or may not be identical to the patterns of the species that contain them (see Fig. 12.7). Suppose that genes g_B and g_C are homologous. They normally will have the same function. Suppose that g_B in species B has undergone a number of duplication events to produce $g_{B1}, g_{B2}, \dots, g_{Bn}$. Selection for function may require that one of these copies (g_{B1} , for example) retains the ancestral function, but then g_{B2}, \dots, g_{Bn} can evolve either to lose function or possibly acquire new but related functions. Sequence homologs from two *different* species that share the same function (g_C and g_{B1} in the example above) are called **orthologs**. Homologs of gene copies that have evolved *within* a species (e.g., any two of g_{B1}, \dots, g_{Bn}) are said to be **paralogs**. Paralogy arises from independent evolutionary events affecting a gene within one species but not the corresponding gene in a related species.

An important activity in biology is identifying DNA or protein sequences that are similar to a sequence of experimental interest, with the goal of finding sequence homologs among a list of *similar* sequences. By writing the sequence of gene g_A and of each candidate homolog as strings of characters, with one string above the other, we can determine at which positions the strings do or do not match. This is called an **alignment**. As we shall see, there are many different ways that two strings might be aligned. Ordinarily, we expect homologs to have more matches than two randomly chosen sequences.

However, this seemingly simple alignment operation is not as simple as it sounds. Consider the examples below (matches are indicated by \cdot , and $-$ is placed opposite bases not aligned):

$$\begin{array}{ll}
 \text{ACGTCTAG} & 2 \text{ matches} \\
 \dots & 5 \text{ mismatches} \\
 \text{ACTCTAG-} & 1 \text{ not aligned}
 \end{array} \tag{6.1}$$

We might instead have written the sequences

$$\begin{array}{ll}
 \text{ACGTCTAG} & 5 \text{ matches} \\
 \dots\dots & 2 \text{ mismatches} \\
 \text{-ACTCTAG} & 1 \text{ not aligned}
 \end{array} \tag{6.2}$$

We might also have written

$$\begin{array}{ll}
 \text{ACGTCTAG} & 7 \text{ matches} \\
 \dots\dots\dots & 0 \text{ mismatches} \\
 \text{AC-TCTAG} & 1 \text{ not aligned}
 \end{array} \tag{6.3}$$

Which alignment is “better”?

Next, consider aligning the sequence TCTAG with a long DNA sequence:

$$\begin{array}{r} \dots \text{AACTGAGTTTACGCTCATAGA} \dots \\ \text{T---CT-A--G} \end{array} \quad (6.4)$$

We might suspect that if we compared any string of modest length with another very long string, we could obtain perfect agreement if we were allowed the option of “not aligning” with a sufficient number of letters in the long string.

Clearly, we would prefer some type of parsimonious alignment—one that does not postulate an excessive number of letters in one string that are not aligned opposite identical letters in the other. We have seen that there are multiple ways of aligning two sequence strings, and we may wish to compare our **target** string (target meaning the given sequence of interest) to entries in databases containing more than 10^7 sequence entries or to collections of sequences billions of letters long. How do we do this? The present chapter focuses on alignment of two sequences with each other. This can be done using the entire strings (**global alignment**) or by looking for shorter regions of similarity contained within the strings that otherwise do not significantly match (**local alignment**). We briefly discuss multiple-sequence alignment (alignment of more than two strings) at the end of this chapter.

Our approach is guided by biology. It is possible for evolutionarily related proteins and nucleic acids to display substitutions at particular positions (resulting from known mutational processes). Also, it is possible for there to be insertions or deletions of sequences (less likely than substitution). In total, the DNA sequence can be modified by (Section 1.3.3):

- Substitution (point mutation)
- Insertion of short segments
- Deletion of short segments
- Segmental duplication
- Inversion
- Transposable element insertion
- Translocation

The latter four processes often involve DNA segments larger than the coding regions of genes, and they usually don’t affect the types of alignment presently under discussion. The first three processes are important in aligning targets whose sizes are less than or equal to the size of coding regions of genes, and these will be explicitly treated in the alignment processes. Insertions and/or deletions are called **indels**. In (6.3) above, we can’t tell whether the string at the top resulted from the insertion of **G** in ancestral sequence **ACTCTAG** or whether the sequence at the bottom resulted from the deletion of **G** from ancestral sequence **ACGTCTAG**. For this reason, alignment of a letter opposite nothing is simply described as an indel.

Before proceeding, note that we may align nucleic acids with each other or polypeptide sequences with each other. The latter case raises a number of additional issues because of constraints on protein structures and the genetic code, so it will be discussed in the next chapter.

6.2 Basic Example

Before proceeding to a rigorous description, we will introduce the “flavor” of the process with a simple “toy” example. Suppose that we wish to align **WHAT** with **WHY**. Our goal is to find the highest-scoring alignment. This means that we will have to devise a scoring system to characterize each possible alignment. One possible alignment solution is

WHAT
WH-Y

We need a rule to tell us how to calculate an alignment score that will, in turn, allow us to identify which alignment is best. Let’s use the following scores for each instance of match, mismatch, or indel:

identity (match)	+1
substitution (mismatch)	$-\mu$
indel	$-\delta$

The minus signs for substitutions and indels assure that alignments with many substitutions or indels will have low scores. We define the score S as the sum of individual scores at each position:

$$S(\text{WHAT}/\text{WH}-\text{Y}) = 1 + 1 - \delta - \mu.$$

There is a more general way of describing the scoring process (not necessary for “toy” problems such as the one above). We write the target sequence (**WHY**) and the search space (**WHAT**) as rows and columns of a matrix:

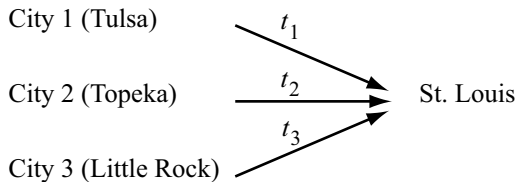
	-	W	H	A	T
-					
W		x			
H			x	x	
Y					x

We have placed an x in the matrix elements corresponding to a particular alignment shown. We have included one additional row and one additional column for initial indels ($-$) to allow for the possibility (not applicable here) that alignments do not start at the initial letters (W opposite W in this case). We can indicate the alignment above as a path through elements of the matrix (arrows). If the sequences being compared were identical, then this path would be along the diagonal. Other alignments of WHAT with WHY would correspond to paths through the matrix other than the one shown. Each step from one matrix element to another corresponds to the incremental shift in position along one or both strings being aligned with each other, and we could write down in each matrix element the running score up to that point instead of inserting x .

	-	W	H	A	T
-	0				
W		1			
H			2	$2-\delta$	
Y					$2-\delta-\mu$

What we seek is the path through the matrix that produces the greatest possible score in the element at the lower right-hand corner. That is our “destination,” and it corresponds to having used up all of the letters in the search string (first column) and search space (first row)—this is the meaning of *global alignment*.

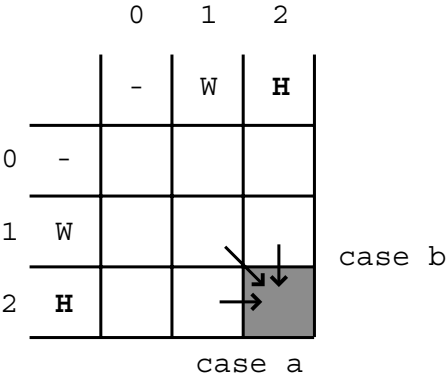
Using a scoring matrix such as this employs a particular trick of thinking. For example, what is the “best” driving route from Los Angeles to St. Louis? We could plan our trip starting in Los Angeles and then proceed city to city considering different routes. For example, we might go through Phoenix, Albuquerque, Amarillo, etc., or we could take a more northerly route through Denver. We seek an itinerary (best route) that minimizes the driving time. One way of analyzing alternative routes is to consider the driving time to a city relatively close to St. Louis and add to it the driving time from that city to St. Louis. Three examples for the last leg of the trip are shown below.



We would recognize that the best route to St. Louis is the route to St. Louis from city n + the best route to n from Los Angeles. If D_1 , D_2 ,

and D_3 are the driving times to cities 1, 2, and 3 from Los Angeles, then the driving times to St. Louis through these cities are given by $D_1 + t_1$, $D_2 + t_2$, and $D_3 + t_3$. Suppose that the driving time to St. Louis through Topeka (City 2) turned out to be smaller than the times to St. Louis through Tulsa or Little Rock (i.e., $D_2 + t_2$ were the minimum of $\{D_1 + t_1, D_2 + t_2, D_3 + t_3\}$). We then know that we should travel through Topeka. We next ask how we get to Topeka from three prior cities (not shown), seeking the route that minimizes the driving time to City 2. Analyzing the best alignment using an **alignment matrix** proceeds similarly, first filling in the matrix by working forward and then working backward from the “destination” (last letters in a global alignment) to the starting point. This general approach to problem-solving is called *dynamic programming*.

The best alignment is revealed by beginning at the destination (lower right-hand corner matrix element) and working backward, identifying the path that maximizes the score at the end. To do this, we will have to calculate scores for all possible paths into each matrix element (“city”) from its neighboring elements above, to the left, and diagonally above. To illustrate, suppose that we want to continue an ongoing alignment process using **WHAT** and **WHY** and that we have gotten to the point at which we want to continue the alignment into the shaded element of the matrix above. We have now added row and column numbers to help us keep track of matrix elements.



There are three possible paths into element (3, 3) (aligning left to right with respect to both strings—letters not yet aligned written in parentheses):

- Case a. If we had aligned **WH** in **WHY** with **W** in **WHAT** (corresponding to element (2, 1)), adding **H** in **WHAT** without aligning it to **H** in **WHY** corresponds to an insertion of **H** (relative to **WHY**) and advances the alignment from element (2, 1) to element (2, 2) (horizontal arrow):

(W)H(AT)
 (WH)- (Y)

- Case b. If we had aligned **W** in **WHY** with **WH** in **WHAT** (corresponding to element (1, 2)), adding the **H** in **WHY** without aligning it to **H** in **WHAT** corre-

sponds to insertion of H (relative to WHAT) and advances the alignment from element (1, 2) to element (2, 2) (vertical arrow):

(WH)–(AT)

(W)H (Y)

Case c. If we had aligned W in WHY with W in WHAT (corresponding to element (1,1)), then we could advance to the next letter in both strings, advancing the alignment from (1, 1) to (2, 2) (diagonal arrow above):

(W)H(AT)

(W)H (Y)

Note that horizontal arrows correspond to adding indels to the string written vertically and that vertical arrows correspond to adding indels to the string written horizontally.

Associated with each matrix element (x, y) from which we could have come into (2, 2) is the score $S_{x,y}$ up to that point. Suppose that we assigned scores based on the following scoring rules:

identity (match)	+1
substitution (mismatch)	–1
indel	–2

Then the scores for the three different routes into (2,2) are

$$\text{Case a : } S_{2,2} = S_{2,1} - 2,$$

$$\text{Case b : } S_{2,2} = S_{1,2} - 2,$$

$$\text{Case c : } S_{2,2} = S_{1,1} + 1.$$

The path of the cases a, b, or c that yields the highest score for $S_{2,2}$ is the preferred one, telling us which of the alignment steps is best.

Using this procedure, we will now go back to our original alignment matrix and fill in all of the scores for all of the elements, *keeping track of the path into each element that yielded the maximum score to that element*. The initial row and column labeled by (–) corresponds to sliding WHAT or WHY incrementally to the left of the other string without aligning against any letter of the other string. Aligning (–) opposite (–) contributes nothing to the alignment of the strings, so element (0,0) is assigned a score of zero. Since penalties for indels are –2, the successive elements to the right or down from element (0,0) each are incremented by –2 compared with the previous one. Thus $S_{0,1}$ is –2, corresponding to W opposite –, $S_{0,2}$ is –4, corresponding to WH opposite –, etc., where the letters are coming from WHAT. Similarly, $S_{1,0}$ is –2, corresponding to – opposite W, $S_{2,0}$ is –4, corresponding to -- opposite WH, etc., where the letters are coming from WHY. The result up to this point is

		0	1	2	3	4
		-	W	H	A	T
0	-	0	-2	-4	-6	-8
1	W	-2				
2	H	-4				
3	Y	-6				

Now we will calculate the score for $(1, 1)$. This is the greatest of $S_{0,0} + 1$ (W matching W, starting from $(0, 0)$), $S_{0,1} - 2$, or $S_{1,0} - 2$. Clearly, $S_{0,0} + 1 = 0 + 1 = 1$ “wins”. (The other sums are $-2 - 2 = -4$.) We record the score value $+1$ in element $(1, 1)$ and record an arrow that indicates where the score came from.

		0	1	2	3	4
		-	W	H	A	T
0	-	0	-2	-4	-6	-8
1	W	-2	+1			
2	H	-4	-1			
3	Y	-6				

The same procedure is used to calculate the score for element $(2, 1)$ (see above). Going from $(1, 0)$ to $(2, 1)$ implies that H from WHY is to be aligned with W of WHAT (after first having aligned W from WHY with $(-)$). This would correspond to a substitution, which contributes -1 to the score. So one possible value of $S_{2,1} = S_{1,0} - 1 = -3$. But $(2, 1)$ could also be reached from $(1, 1)$, which corresponds to aligning H in WHY opposite an indel in WHAT (i.e., not advancing a letter in WHAT). From that direction, $S_{2,1} = S_{1,1} - 2 = 1 - 2 = -1$. Finally, $(2, 1)$ could be entered from $(2, 0)$, corresponding to aligning W in WHAT with an indel coming after H in WHY. In that direction, $S_{2,1} = S_{2,0} - 2 = -4 - 2 = -6$. We record the maximum score into this cell ($S_{2,1} = S_{1,1} - 2 = -1$) and the direction from which it came.

The remaining elements of the matrix are filled in by the same procedure, with the following result:

		0	1	2	3	4
	-	0	W	H	A	T
0	-	0	-2	-4	-6	-8
1	W	-2	+1	1	-3	-5
2	H	-4	-1	2	0	-2
3	Y	-6	-3	0	1	-1

The final score for the alignment is $S_{3,4} = -1$. The score could have been achieved by either of two paths (implied by two arrows into (3, 4) yielding the same score). The path through element (2, 3) (upper path, bold arrows) corresponds to the alignment

WHAT

WH-Y

which is read by tracing back through all of the elements visited in that path. The lower path (through element (3, 3)) corresponds to the alignment

WHAT

WHY-

Each of these alignments is equally good (two matches, one mismatch, one indel).

Note that we always recorded the score for the best path into each element. There are paths through the matrix corresponding to very “bad” alignments. For example, the alignment corresponding to moving left to right along the first row and then down the last column is

WHAT ---

--- WHY

with score -14 .

For this simple problem, it was unnecessary to go through all of these operations. But when the problems get bigger, there are so many different possible alignments that an organized approach such as the one shown here is essential. Biologically interesting alignment problems are far beyond what we can handle with a No. 2 pencil and a sheet of paper.

6.3 Global Alignment: Formal Development

We will now recapitulate the development described in the previous section in a more direct and formal manner. We are given two strings, not necessarily of the same length, from the same alphabet:

$$\begin{aligned} A &= a_1 a_2 a_3 \cdots a_n, \\ B &= b_1 b_2 b_3 \cdots b_m. \end{aligned}$$

Alignment of these strings corresponds to consecutively selecting each letter or inserting an indel in the first string and matching that particular letter or indel with a letter in the other string, or introducing an indel in the second string to place opposite a letter in the first string. Graphically, the process is represented by using a matrix as shown below for $n = 3$ and $m = 4$:

	0	1	2	3	4	
		-	b_1	b_2	b_3	b_4
0	-	→				
1	a_1		↘	→		
2	a_2				↘	
3	a_3					↓

The alignment corresponding to the path indicated by the arrows is

$$\begin{array}{cccccc} b_1 & b_2 & b_3 & b_4 & - & \\ - & a_1 & - & a_2 & a_3 & \end{array} \quad (6.5)$$

Any alignment that can be written corresponds to a unique path through the matrix. The quality of an alignment between A and B is measured by a score, $S(A, B)$, which is large when A and B have a high degree of similarity. If letters a_i and b_j are aligned opposite each other and are the same, they are an instance of an **identity**. If they are different, they are said to be a **mismatch**. The score for aligning the first i letters of A with the first j letters of B is

$$S_{i,j} = S(a_1 a_2 \cdots a_i, b_1 b_2 \cdots b_j). \quad (6.6)$$

$S_{i,j}$ is computed recursively as follows. There are three different ways that the alignment of $a_1 a_2 \cdots a_i$ with $b_1 b_2 \cdots b_j$ can end:

$$\begin{array}{ll}
\text{Case a:} & (a_1 \ a_2 \ \cdots \ a_i) \ - \\
& (b_1 \ b_2 \ \cdots \ b_{j-1}) \ b_j, \\
\text{Case b:} & (a_1 \ a_2 \ \cdots \ a_{i-1}) \ a_i \\
& (b_1 \ b_2 \ \cdots \ b_j) \ -, \\
\text{Case c:} & (a_1 \ a_2 \ \cdots \ a_{i-1}) \ a_i \\
& (b_1 \ b_2 \ \cdots \ b_{j-1}) \ b_j,
\end{array} \tag{6.7}$$

where the inserted spaces “-” correspond to insertions or deletions (“indels”) in A or B . Scores for each case are defined as follows:

$$s(a_i, b_j) = \text{score of aligning } a_i \text{ with } b_j \tag{6.8}$$

$$(\text{= } +1 \text{ if } a_i = b_j, -\mu \leq 0 \text{ if } a_i \neq b_j, \text{ for example}),$$

$$s(a_i, -) = s(-, b_j) = -\delta \leq 0 \text{ (for indels)}. \tag{6.9}$$

With global alignment, indels will be added as needed to one or both sequences such that the resulting sequences (with indels) have the same length. The best alignment up to positions i and j corresponds to the case a, b, or c in (6.7) that produces the largest score for $S_{i,j}$:

$$S_{i,j} = \max \left\{ \begin{array}{l} S_{i-1,j-1} + s(a_i, b_i) \\ S_{i-1,j} - \delta \\ S_{i,j-1} - \delta \end{array} \right\} \begin{array}{l} \text{Case c} \\ \text{Case b.} \\ \text{Case a} \end{array} \tag{6.10}$$

The “max” indicates that the one of the three expressions that yields the maximum value will be employed to calculate $S_{i,j}$. Except for the first row and first column in the alignment matrix, the score at each matrix element is to be determined with the aid of the scores in the elements immediately above, immediately to the left, or diagonally above and to the left of that element, as indicated in (6.10). The scores for elements in the first row and column of the alignment matrix are given by

$$S_{i,0} = -i\delta, \quad S_{0,j} = -j\delta. \tag{6.11}$$

The score for the best global alignment of A with B is $S(A, B) = S_{n,m}$, and it corresponds to the highest-scoring path through the matrix and ending at element (n, m) . It is determined by tracing back element by element along the path that yielded the maximum score into each matrix element.

Computational Example 6.1: Alignment scores

What is the maximum score and corresponding alignment for aligning $A = \text{ATCGT}$ with $B = \text{TGGTG}$? For scoring, take $s(a_i, b_j) = +1$ if $a_i = b_j$, $s(a_i, b_j) = -1$ if $a_i \neq b_j$, and $s(a_i, -) = s(-, b_j) = -2$.

Step 1: Write down the alignment matrix using B along the top and A in a column at the side.

Step 2: Fill in the first row and first column by using (6.11).

	0	1	2	3	4	5	
	-	T	G	G	T	G	
0	-	0	-2	-4	-6	-8	-10
1	A	-2					
2	T	-4					
3	C	-6					
4	G	-8					
5	T	-10					

Step 3: Fill in all matrix elements using the scoring rules of (6.10) and keeping track of the path into each element that was employed. Sometimes two paths will yield equal scores.

	0	1	2	3	4	5	
	-	T	G	G	T	G	
0	-	0	-2	-4	-6	-8	-10
1	A	-2	-1	-3	-5	-7	-9
2	T	-4	-1	-2	-4	-4	-6
3	C	-6	-3	-2	-3	-5	-5
4	G	-8	-5	-2	-1	-3	-4
5	T	-10	-7	-4	-3	0	-2

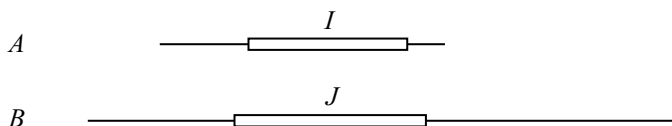
Step 4: Read out the alignment, starting at (5, 5) and working backward. For clarity, arrows corresponding to the path for the highest-scoring alignment are drawn with heavier lines.

The matrix elements corresponding to the last three steps in the alignment below are shown by corresponding shading.

A: A T C G T -
 B: - T G G T G

6.4 Local Alignment: Rationale and Formulation

Often proteins are multifunctional. Pairs of proteins that share one of these functions may have regions of similarity embedded in otherwise dissimilar sequences. For example, human TGF- β receptor (which we will label A) is a 503 amino acid (aa) residue protein containing a protein kinase domain extending from residue 205 through residue 495. This 291 aa residue segment of TGF- β receptor is similar to an interior 300 aa residue portion of human bone morphogenic protein receptor type II precursor (which we will label B), a polypeptide that is 1038 aa residues long. This type of situation is diagrammed as



where regions of similarity I, J are indicated by the boxes. With only partial sequence similarity and very different lengths, attempts at global alignment of two sequences such as these would lead to huge cumulative indel penalties. What we need is a method to produce the best *local alignment*; that is, an alignment of segments contained *within* two strings (Smith and Waterman, 1981).

As before, we employ an alignment matrix, and we seek a high-scoring path through the matrix. However, this time the path will traverse only *part* of the matrix. Also, we do not apply indel penalties if strings A and B fail to align at the ends. This means that instead of having elements $-i\delta$ and $-j\delta$ in the first row and first column, respectively ($-\delta$ being the penalty for each indel), all the elements in the first row and first column will now be zero. Moreover, since we are interested in paths that yield high scores over stretches less than or equal to the length of the smallest string, there is no need to continue paths whose scores become too small. Therefore, if the best path to an element from its immediate neighbors above and to the left (including the diagonal) leads to a negative score, we will arbitrarily assign a 0 score to that element. We will identify the best local alignment by tracing back from the matrix element having the highest score. This is usually not (but occasionally may be) the element in the lower right-hand corner of the matrix.

The mathematical statement of the problem is as follows. We are given two strings $A = a_1a_2a_3 \dots a_n$ and $B = b_1b_2b_3 \dots b_m$. Within each string there are intervals I and J that have similar sequences. I and J are *intervals* of A and B , respectively. We indicate this by writing $I \subset A$ and $J \subset B$, where “ \subset ” means “is an interval of.” The best local alignment score, $M(A, B)$, for strings A and B is

$$M(A, B) = \max\{S(I, J) : I \subset A, J \subset B\} \quad (6.12)$$

where $S(I, J)$ is the score for subsequences I and J and $S(\emptyset, \emptyset) = 0$. Elements of the alignment matrix are $M_{i,j}$, and since we are not applying indel penalties at the ends of A and B , we write

$$M_{i,0} = M_{0,j} = 0. \quad (6.13)$$

The score up to and including the matrix element $M_{i,j}$ is calculated by using scores for the elements immediately above and to the left (including the diagonal), but this time scores that fall below zero will be replaced by zero. The scoring for matches, mismatches, and indels is otherwise the same as for global alignment. The resulting expression for scoring $M_{i,j}$ is

$$M_{i,j} = \max \left\{ \begin{array}{l} M_{i-1,j-1} + s(a_i, b_j) \\ M_{i-1,j} - \delta \\ M_{i,j-1} - \delta \\ 0 \end{array} \right\}. \quad (6.14)$$

The best local alignment is the one that ends in the matrix element having the highest score:

$$\max\{S(I, J) : I \subset A, J \subset B\} = \max_{i,j} M_{i,j}. \quad (6.15)$$

Thus, the best local alignment score for strings A and B is

$$M(A, B) = \max_{i,j} M_{i,j}. \quad (6.16)$$

Computational Example 6.2: Local alignment

Determine the best local alignment and the maximum alignment score for $A = \text{ACCTAAGG}$ and $B = \text{GGCTCAATCA}$. For scoring, take $s(a_i, b_j) = +2$ if $a_i = b_j$, $s(a_i, b_j) = -1$, $a_i \neq b_j$, and $s(a_i, -) = s(-, b_j) = -2$.

- Step 1: Write down the alignment matrix using B along the top and A in a column at the side.
- Step 2: Fill in the first row and first column by using (6.13).
- Step 3: Then fill in all matrix elements using the scoring rule (6.14), keeping track of the paths into each element. For clarity, we have included below only the arrows around the highest-scoring path. Observe what happens for $M_{3,7}$. Regardless of whether this is entered from above, from the left, or diagonally from the left, the scoring rule would have yielded -1 were it not for the requirement that all elements be non-negative, as indicated in (6.14).
- Step 4: The local alignment ends at $M_{7,9}$ (shaded box), which contains the maximum alignment score (6). Read out the alignment, starting at $M_{7,9}$ and working backward along the directions of entry into each cell until an element containing zero is encountered.

	0	1	2	3	4	5	6	7	8	9	10	
	-	G	G	C	T	C	A	A	T	C	A	
0	-	0	0	0	0	0	0	0	0	0	0	
1	A	0	0	0	0	0	0	2	2	0	0	2
2	C	0	0	0	2	0	2	0	1	1	2	0
3	C	0	0	0	2	1	2	1	0	0	3	1
4	T	0	0	0	0	4	2	1	0	2	1	2
5	A	0	0	0	0	2	3	4	3	2	1	3
6	A	0	0	0	0	0	1	5	6	4	2	3
7	G	0	2	2	0	0	0	3	4	5	3	1
8	G	0	2	4	2	0	0	1	2	3	4	2

The resulting local alignment is enclosed in the box below:

A:	A C	C T - A A	G G -
B:	G G	C T C A A	T C A

Most local alignment programs only report the aligned regions of A and B , that is, the sequences shown in the box above.

6.5 Number of Possible Global Alignments

We have shown how to identify rigorously the highest-scoring alignment. Obviously, for strings of lengths n and m , we had to compute three scores going into $(n-1)(m-1)$ cells and to take a maximum. This means that $4nm$ computations were required (including the trivial first row and first column). In the “big O ” notation (Section 4.3), this means that computation time will be $O(mn)$.

Now we ask a separate question: How many *possible* global alignments are there for two strings of lengths m and n ? This is the same thing as asking how many different paths there are through the alignment matrix (ex-

cluding backward moves along the strings). We recognize that any alignment that has any practical meaning will have matched some of the letters in one string with some of the letters in the other. The number of matched pairs will be less than or equal to the smaller of m and n . We can count the number of alignments, $\#A$, by summing the number of alignments having one matched pair, the number of alignments having two matched pairs, and so on up to $\min(m, n)$ matched pairs. Examples of some of the 12 alignments of $A = a_1a_2a_3a_4$ and $B = b_1b_2b_3$ having one matched pair are shown below.

$$\begin{array}{cccccccc} - & - & a_1 & a_2 & a_3 & a_4 & & & - & a_1 & a_2 & a_3 & a_4 & - & & - & a_1 & - & a_2 & a_3 & a_4 \\ b_1 & b_2 & b_3 & - & - & - & & & b_1 & b_2 & - & - & - & b_3 & & & b_1 & - & b_2 & - & - & b_3 \end{array}$$

The number of alignments, $\#A$, is the sum of the numbers of alignments having 1, 2, 3, . . . and $\min(m, n)$ matched pairs. To count the number of ways of having k aligned pairs, we must choose k letters from each sequence. From A this can be done in $\binom{n}{k}$ ways, and from B this can be done in $\binom{m}{k}$ ways. Therefore

$$\begin{aligned} \#A &= \sum_{k=0}^{\min(m, n)} (\# \text{ of alignments having exactly } k \text{ matched pairs}) \\ &= \sum_{k=0}^{\min\{m, n\}} \binom{n}{k} \binom{m}{k} \\ &= 1 + nm + \frac{n!}{(n-2)!2!} \times \frac{m!}{(m-2)!2!} + \dots \end{aligned} \tag{6.17}$$

The “1” is the number of ways of choosing no letters from n letters and m letters, nm is the number of ways of choosing one letter from n letters and one letter from m letters (the number of alignments having one matched pair), etc. The result turns out to have a simple expression:

$$\#A = \sum_{k=0}^{\min\{m, n\}} \binom{n}{k} \binom{m}{k} = \binom{n+m}{\min(n, m)}. \tag{6.18}$$

The latter equality requires some manipulation, which is provided at the end of this chapter. For the special case for which $m = n$,

$$\#A = \binom{2n}{n} = (2n)!/(n!)^2. \tag{6.19}$$

(6.19) can be approximated by using Stirling’s approximation

$$x! \sim (2\pi)^{1/2} x^{(x+1/2)} e^{-x}.$$

When this is applied to (6.19), we obtain

$$\#A \sim 2^{2n} / \sqrt{\pi n}. \quad (6.20)$$

This approximate value for the number of alignments can also be rationalized in the following simple manner. We are given two strings of equal length, $A = a_1 a_2 \dots a_n$ and $B = b_1 b_2 \dots b_n$. For each of the letters in A , we have two choices: align it opposite a letter in B or add an indel. This makes 2^n ways of handling the letters in A . Similarly, for B there are two ways of handling each letter, for a total of 2^n ways of handling the letters in B . Since the decision “align or add indel” is made for every letter in both strings, the total number of choices for both strings (the approximate number of alignments) is $2^n \times 2^n = 2^{2n}$.

For a simple problem of $A = a_1 a_2 a_3 a_4$ aligned with $B = b_1 b_2 b_3$, $n + m = 7$, $\min(m, n) = 3$, and the exact number of alignments is

$$\#A = \binom{n+m}{m} = \binom{4+3}{3} = 7! / (4! 3!) = 35. \quad (6.21)$$

For longer strings, the number of alignments gets very large very rapidly. For $n = m = 10$, the number of alignments is already 184,756. For $n = m = 20$, the number of alignments is 1.38×10^{11} . For $m = n = 100$, there are $\sim 2^{200}$ possible alignments. In more familiar terms (using $\log_{10} x = \log_2 x \times \log_{10} 2$), $\log_{10}(2^{200}) = \log_2(2^{200}) \times \log_{10}(2) = 200 \times (0.301) \approx 60$. In other words, $\#A$ when aligning two strings of length 100 is about 10^{60} . This is an astronomically large number. For example, the Sun weighs 1.99×10^{33} grams. Each gram contains roughly 12×10^{23} protons and electrons, which means that the Sun contains about 24×10^{56} elementary particles. It would take 400 stars the size of our Sun to contain as many elementary particles as there are alignments between two equal-length strings containing 100 characters.

Clearly, we need to have ways of further simplifying the alignment process beyond our $O(nm)$ method, and this is the topic of the next chapter.

Proof of equality (6.18)

The hypergeometric distribution pertains to the sampling (without replacement) of a binary population. If there are m “successes” in a population of size N , the probability of drawing k successes in a sample of size h is given by

$$g(k; h, m, N) = \binom{m}{k} \binom{N-m}{h-k} / \binom{N}{h}.$$

This is called the hypergeometric distribution. In our situation, the population is the total number of letters that must appear in the global alignment, $N = m + n$. Therefore,

$$g(k; h, m, m+n) = \binom{m}{k} \binom{n}{h-k} / \binom{m+n}{h}.$$

This also holds in particular for $h = n$:

$$g(k; n, m, m+n) = \binom{m}{k} \binom{n}{n-k} / \binom{m+n}{n}.$$

Since $g(k; n, m, m+n)$ is the probability for a particular value of k , summing over all values of k will yield unity:

$$\sum_k g(k; n, m, m+n) = \sum_k \binom{m}{k} \binom{n}{n-k} / \binom{m+n}{n} = 1.$$

From the definition of $\binom{n}{k}$, it is clear that $\binom{n}{n-k} = \binom{n}{k}$. Thus,

$$\sum_k \binom{m}{k} \binom{n}{k} = \binom{m+n}{n},$$

which is the right-hand side of (6.18) when $n = \min(m, n)$.

6.6 Scoring Rules

We have used alignments of nucleic acids as examples, but one very important application of alignment is protein alignment, for which scoring is much more complicated. At this point, we address briefly the issue of assigning appropriate values to $s(a_i, b_j)$, $s(a_i, -)$, and $s(-, b_j)$ for nucleotides. We address the scoring for amino acids in the next chapter.

Considering $s(a_i, b_j)$ first, we write down a **scoring matrix** containing all possible ways of matching a_i with b_j , $a_i, b_j \in \{\mathbf{A}, \mathbf{C}, \mathbf{G}, \mathbf{T}\}$ and write in each element the scores that we have used for matches and mismatches in the examples above.

		$b_j :$			
		A	C	G	T
$a_i :$	A	1	-1	-1	-1
	C	-1	1	-1	-1
	G	-1	-1	1	-1
	T	-1	-1	-1	1

This scoring matrix contains the assumption that aligning A with G is just as bad as aligning A with T because the mismatch penalties are the same in

both cases. However, studies of mutations in homologous genes indicate that **transition** mutations ($A \rightarrow G$, $G \rightarrow A$, $C \rightarrow T$, or $T \rightarrow C$) occur approximately twice as frequently as do **transversions** ($A \rightarrow T$, $T \rightarrow A$, $A \rightarrow C$, $G \rightarrow T$, etc.). Therefore, it may make sense to apply a lesser penalty for transitions than for transversions since they are more likely to occur (i.e., related sequences are more likely to have transitions, so we should not penalize transitions as much). The collection of $s(a_i, b_j)$ values in that case might be represented in the matrix below:

		$b_j :$			
		A	C	G	T
$a_i :$	A	1	-1	-0.5	-1
	C	-1	1	-1	-0.5
	G	-0.5	-1	1	-1
	T	-1	-0.5	-1	1

A second issue relates to the scoring of gaps (a succession of indels). Are indels independent? Up to now, we have scored a gap of length k (see (6.11)) as

$$w(k) = -k\delta. \quad (6.22)$$

However, insertions and deletions sometimes appear in “chunks” as a result of biochemical processes such as replication slippage at microsatellite repeats. Also, deletions of one or two nucleotides in protein-coding regions would produce frameshift mutations (usually nonfunctional), but natural selection might allow small deletions that are integral multiples of 3, which would preserve the reading frame and some degree of function. These examples suggest that it would be better to have gap penalties that are not simply multiples of the number of indels. One approach is to use an expression such as

$$w(k) = -\alpha - \beta(k - 1). \quad (6.23)$$

This would allow one to impose a larger penalty for opening a gap ($-\alpha$) and a smaller penalty for gap extension ($-\beta$ for each additional base in the gap).

6.7 Multiple Alignment

Now that we have explained methods for aligning two sequences, the issue of aligning three or more sequences naturally arises. It might be that we

have many sequences of orthologous genes from different organisms, and we are interested in the relationships between these sequences. Sometimes genes within an organism arise from duplications and the history of the duplications (along with the associated functions) must be inferred. How are the methods we have constructed in this chapter generalized?

It turns out to be almost obvious how to generalize the dynamic programming methods. We will indicate how this is done for three sequences. The basic idea in the pairwise case was to have a recursion for the end of the alignment. Recall that $A = a_1a_2 \dots a_i$ and $B = b_1b_2 \dots b_j$ have alignments that can end in three possibilities: $(-, b_j)$, (a_i, b_j) , or $(a_i, -)$. In other words, the alignments end in an indel or aligned letters. The number of alternatives for aligning a_i with b_j can be calculated as $3 = 2^2 - 1$, which can be understood as follows. There are two things that you can do at the i th position of sequence A : align or employ an indel. There are two things that you can do at the j th position of sequence B : align or employ an indel. Therefore there are 2^2 alternatives for two sequences. However, “align” is the same event for both sequences, so we subtract 1 from 2^2 to obtain the number of distinct combinations. If we now introduce a third sequence $c_1c_2 \dots c_k$, the three-sequence alignment can end in one of seven ways ($7 = 2^3 - 1$): $(a_i, -, -)$, $(-, b_j, -)$, $(-, -, c_k)$, $(-, b_j, c_k)$, $(a_i, -, c_k)$, $(a_i, b_j, -)$, and (a_i, b_j, c_k) . In other words, the alignment ends in 0, 1, or 2 indels. This fundamental term in the recursion is no problem, except that it must be done in time and space proportional to the number of (i, j, k) positions; that is the product of the length of the sequences $i \times j \times k$. This dramatic increase in running time and storage requirements continues as we increase the number of sequences, so that this method is practical for small problems only. Hence the bioinformatics community has found heuristic solutions.

One class of methods for speeding up the calculation employs pairwise alignments in an incremental fashion: the most similar pair is placed into a fixed alignment, and then the other sequences are included in a stepwise fashion. One of the most used and practical programs to perform global multiple alignment is CLUSTALW (Thompson et al., 1994). That program proceeds by computing all pairwise alignments, estimating a tree (or cluster) of relationships using those alignment scores (see Section 10 for a discussion of clustering), and then collecting the sequences into a multiple alignment using the tree and pairwise alignments as a guide for adding each successive sequence.

Another approach developed in recent years uses hidden Markov models (HMMs) and has proven quite effective (see, e.g., www.cse.usc.edu/research/compbio/sum.html).

6.8 Implementation

We used short sequences in the illustrations to keep the computations minimal. For real problems with realistically long sequences (hundreds of amino acid residues to thousands of base pairs long), substantial computational resources may be required. In fact, there are commercially available computers containing application-specific integrated circuits specifically tailored for dynamic programming applications. The Smith-Waterman pairwise local alignment is available in a number of commercial software packages and at various Internet sites. One example is the MPsrch tool from the European Bioinformatics Institute (<http://www.ebi.ac.uk/MPsrch>).

The computations for global and local alignments are presented in pseudocode in Computational Example 6.3. Although the code looks simple, remember that these computations may require large amounts of memory for numerous operations. Consequently, alignment that relies exclusively on dynamic programming approaches tends to be slow. In the next chapter, we will discuss alignment methods that are less accurate but much faster.

Computational Example 6.3: Pseudocode for global and local alignment.

Global alignment

```

Input sequences A, B
Set  $S_{i,0} \leftarrow -\delta i$  for all  $i$ 
Set  $S_{0,j} \leftarrow -\delta j$  for all  $j$ 
for  $i = 1$  to  $n$ 
     $j = 1$  to  $m$ 
         $S_{i,j} \leftarrow \max\{S_{i-1,j} - \delta, S_{i-1,j-1} + s(a_i, b_j), S_{i,j-1} - \delta\}$ 
    end
end

```

Local alignment

```

Input sequences A, B
Set  $M_{i,0} = M_{0,j} = 0$  for all  $i, j$ 
for  $i = 1$  to  $n$ 
     $j = 1$  to  $m$ 
         $M_{i,j} \leftarrow \max\{M_{i-1,j} - \delta, M_{i-1,j-1} + s(a_i, b_j), M_{i,j-1} - \delta, 0\}$ 
    end
end

```

References

- Smith TF, Waterman MS (1981) The identification of common molecular subsequences. *Journal of Molecular Biology* 147:195–197.
- Thompson JD, Higgins DG, Gibson TJ (1994) CLUSTAL W: Improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. *Nucleic Acids Research* 22:4673–4680.

Exercises

Exercise 1. Another approach to counting alignments is from the recursion formula $B(n, m) = B(n-1, m) + B(n, m-1) + B(n-1, m-1)$, where $B(n, m)$ is the number of alignments of $a_1a_2 \cdots a_n$ with $b_1b_2 \cdots b_m$. Using $B(0, 0) = 0$ and $B(i, 0) = 1 = B(0, j)$ (all i and j), use a matrix to find $B(5, 5)$. Compare this number with

$$\#A = \binom{10}{5}.$$

Why is $B(5, 5)$ larger?

Exercise 2. Using Stirling's approximation, we found an approximation for $\#A$, the number of alignments. Apply that formula to $n = m = 1000$. Why didn't we simply compute

$$\binom{2000}{1000}$$

exactly on our computer?

Exercise 3. If we use a gap penalty $g(k) = -10k$, we observe that all optimal alignments of strings A and B are identical for all values of the mismatch penalty μ when $\mu > 20$. Explain why this is true.

Exercise 4. A score for global alignment of $a_1a_2 \cdots a_n$ with $b_1b_2 \cdots b_m$ is $E - \mu F - \delta G$, where E is the number of matches, F is the number of mismatches, and G is the number of deleted letters. Evaluate this expression for $\mu = -1$ and $\delta = -1/2$.

Exercise 5. Equation (6.10) gives the recursion equation for global alignment of two sequences. Generalize it to an equation for global alignment of three sequences. You will require a function $s(a, b, c)$ where any of the letters can be a blank or “-”.

Exercise 6. Go to the alignment server at <http://www.cmb.usc.edu> and perform local sequence alignment of the following two sequences:

```
>sequence1
TCAGTTGCCAAACCCGCT
>sequence2
AGGGTTGACATCCGTTTT
```

- First set $\alpha = \beta = 1000$, and perform the alignment using the **Single** score option, with **Match** score = 10, and with **Mismatch Penalty** score = 10, 7, 5, or 3. Observe the effect of reducing the Penalty score by examining the ten highest-scoring alignments for each case. What trend do you observe? Explain this trend.
- Now with **Match** score = 10 and **Mismatch Penalty** score = 10, explore the effect of changing α from 15 to 10, and then to 5 (holding β at the default setting of 3). What trend do you observe? Explain this trend.

Exercise 7. Using the alignment server at <http://www.cmb.usc.edu>, perform global sequence alignment of the following two sequences:

```
>sequence1
TCAGTTGCCA
>sequence2
AGGGTTGACA
```

Use **Match** score = 10, **Mismatch Penalty** score = 10, $\alpha = 15$, and $\beta = 3$. Repeat the alignment using the same **Match** score and **Mismatch Penalty** score as above, but with $\alpha = \beta = 10$.

Exercise 8. For the same sequences used in Exercise 7, calculate the alignment matrix with traceback using $\alpha = \beta = 5$, **Match** score = 10, and **Mismatch Penalty** score = 10. Compare your optimal alignments with those from Exercise 7.

Exercise 9. Download sequences contained in accession numbers P21189 and NP_143776.1 from the NCBI GenBank database. Using the alignment server available at <http://www.cmb.usc.edu>, perform the following alignments.

- Perform global alignment using “blosum62” as the penalty matrix (see Section 7.5.2) and the default indel settings ($\alpha = 15$, $\beta = 3$). Examine your result: Can you discern a region that likely will produce a high-scoring local alignment?
- Perform local alignment on the same sequences. Did the result from local alignment agree with your prediction based upon the global alignment?
- Use the contiguous sequence from 1Q8I (remove indels) that resulted from the local alignment with NP_143776.1 as query for an NCBI BLAST search (Chapter 7) of the non-redundant databases. Are any putative conserved domains found? Check your output to see whether there are significant hits to *Danio* or *Arabidopsis* sequences.

Exercise 10. Download gene sequences for histone H2a contained in accession numbers Z48612 (*S.cerevisiae*) and BC001193.1 (*H. sapiens*) from the NCBI GenBank database. (In the case of Z48612, use the “Find” function of your browser to find the annotated gene for H2a, and then click on the “Gene” link.) Supply these sequences in FASTA format to the alignment server available at <http://www.cmb.usc.edu>.

- a. Perform a global alignment, using “Single Score” for type of scoring and default parameters otherwise. Examine the result, and try to identify a region that you would predict to have the highest-scoring local alignment.
- b. Now perform local alignment using the same two sequences and the same parameters. Were you able to predict the region with high-scoring local alignment from the global alignment output? Why or why not?

Rapid Alignment Methods: FASTA and BLAST

7.1 The Biological Problem

In the last chapter, we indicated how alignment could be performed rigorously and some of the reasons for performing it. In this chapter, we consider the practicalities of the alignment process, and we demonstrate how it can be speeded up. The need for accelerated methods of alignment is connected with the potentially large number of possible alignments between two sequences (Section 6.5) and with the very large sizes of the databases that must be searched. Why is it necessary to search large databases?

Remember that alignment involves a **query** or **target** sequence and a **search space**. The query sequence typically comes from the organism that is under investigation. The investigator will have obtained the sequence of a portion of the genome and usually seeks information about its possible function either by direct experimentation or by comparing this sequence with related sequences in other organisms. Direct experimentation on a gene of interest (query) in an arbitrary organism may be difficult for a number of reasons. For example, some organisms are difficult to grow in the laboratory (such as certain types of marine bacteria). Other organisms can be grown but may have little genetic or biochemical data (such as the nine-banded armadillos, which can serve as an animal model for leprosy). Or, there may be organisms that are experimentally refractory (we can't perform arbitrary genetic crosses with *Homo sapiens*, for example) or expensive to work with (such as chimpanzees, which are endangered, expensive, and have relatively long generation times).

A solution to this type of problem is to seek comparisons with genes from a number of **model organisms**—organisms chosen for intensive genomic, genetic, or biochemical studies (Section 1.1). Examples of traditional model organisms are *Escherichia coli* (a bacterium), *Saccharomyces cerevisiae* (baker's yeast), *Caenorhabditis elegans* (a nematode “worm”), *Drosophila melanogaster* (fruit fly), *Arabidopsis thaliana* (a flowering plant—“mustard weed”), and *Mus musculus* (the common mouse). Others are being added to

the list as sequencing projects expand. Because of extensive genetic and biochemical study over several decades, many of the genes, gene products, and functions of gene products are known for these model organisms. Because of the evolutionary relationships among organisms, we ordinarily expect that a gene from other experimental organisms may have homologs in the genomes of one or more model organisms. For example, the homeotic genes that act during the development of human embryos have homologs with *Drosophila* homeotic genes, and in some cases these genes have the same functions. The important point is that a target gene is likely to have a function similar or related to functions of a homolog in one or more model organisms. This means that we can (judiciously) attach to the target gene part of the functional annotation associated with homologs in model organisms.

The problem is to search in protein or DNA sequence databases for sequences that match (to a significant degree) the sequence of interest. These databases contain many entries and large numbers of letters. For example, at the time this book was written, there were over 2 million nonredundant entries accessible using the BLAST server at NCBI (Appendix B), and these contained over 10^{10} letters. In recent years, the amount of DNA sequence in databases has been growing exponentially. From the discussion in the previous chapter, we can readily see that the rigorous alignment method described in the previous chapter is too demanding in memory and computation time for routine searching of large databases: the time to compute an alignment between a string of length n and a string of length m was seen to be proportional to $n \times m$. We need something faster.

So far, we have been talking about performing an alignment between a single query sequence and sequences in databases. What happens if we are performing a whole-genome shotgun sequence assembly of a eukaryotic genome? We'll present more about shotgun sequencing later, but for now we need to know that typically the sequencing includes "reads" of about 500 bp from both ends of each insert in a small plasmid library (insert size 1–3 kb). Typically, enough clones are generated to cover the genome $5\times$ or more. So for a mammal having 3×10^9 bp in its genome, $1\times$ coverage by plasmids with 3 kb inserts would involve 10^6 clones, and $5\times$ coverage would involve 5×10^6 clones. With two sequence reads per clone, the total number of sequence reads is 10^7 . To look for overlaps during sequence assembly, every read would, in principle, need to be compared with (aligned with) every other read. For N reads, there are $N(N - 1)/2$ pairwise comparisons. This means that there are 5×10^{13} comparisons to perform, each of which would require $4 \times (500)^2$ computations if done by dynamic programming. In this case, rapid methods are necessary because of the very large numbers of comparisons that must be made (in principle, any sequence "read" against every other sequence read and its complement).

7.2 Search Strategies

One way to speed up sequence comparison is by reducing the number of sequences to which any candidate sequence must be compared. This can be done by restricting the search for a particular matching sequence to “likely” sequence entries. The logic of the overall process is easily understood if we visualize each sequence to be analyzed as a book in an uncataloged library. Given a book like this one, how could we tell what books are similar just based on word content? The present book has words such as “probabilistic,” “genome,” “statistics,” “composition,” and “distribution.” If we picked up a book at random from an uncataloged library and did not find these words (as might be the case if we had picked books by Jane Austen or Moses Maimonides), we would know immediately that there is no need to search further in *Sense and Sensibility* or *The Guide of the Perplexed* for help in computational biology.

We can reduce the search space by analyzing word content (see Section 3.6). Suppose that we have the query string I indicated below:

I : TGATGATGAAGACATCAG

This can be broken down into its constituent set of overlapping k -tuples. For $k = 8$, this set is

TGATGATG
 GATGATGA
 ATGATGAA
 TGATGAAG
 ·
 ·
 ·
 GACATCAG

If a string is of length n , then there are $n - k + 1$ k -tuples that are produced from the string. If we are comparing string I to another string J (similarly broken down into words), the absence of any one of these words is sufficient to indicate that the strings are not identical. If I and J do not have at least some words in common, then we can decide that the strings are not similar.

We already know that when $\mathbb{P}(A) = \mathbb{P}(C) = \mathbb{P}(G) = \mathbb{P}(T) = 0.25$, the probability that an octamer beginning at any position in string J will correspond to a particular octamer in the list above is $1/4^8$. Provided that J is short, this is not very probable, but if J is long, then it is quite likely that one of the eight-letter words in I can be found in J by chance. Therefore, the appearance of a subset of these words is a *necessary but not sufficient condition* for declaring that I and J have at least some sequence similarity.

7.2.1 Word Lists and Comparison by Content

Rather than scanning each sequence for each k -word, there is a way to collect the k -word information in a set of lists. A list will be a row of a table, where the table has 4^k rows, each of which corresponds to one k -word. For example, with $k = 2$ and the sequences below,

$$\begin{aligned} J &= \text{C C A T C G C C A T C G} \\ I &= \text{G C A T C G G C} \end{aligned}$$

we obtain the word lists shown in Table 7.2. Thinking of the rows as k -words, we denote the list of positions in the row corresponding to the word w as $L_w(J)$ (e.g., with $w = \text{CG}$, $L_{\text{CG}}(J) = \{5, 11\}$). These tables are sparse, since the sequences are short, but serve to illustrate our methods. They can be constructed in a time proportional to the sum of the sequence lengths.

One approach to speeding up comparison is to limit detailed comparisons only to those sequences that share enough “content,” by which we mean k -letter words in common. The statistic that counts k -words in common is

$$\sum_{i=1}^{n-k+1} \sum_{j=1}^{m-k+1} X_{i,j},$$

where $X_{i,j} = 1$ if $I_i I_{i+1} \dots I_{i+k-1} = J_j J_{j+1} \dots J_{j+k-1}$ and 0 otherwise. The computation time is proportional to $n \times m$, the product of the sequence lengths. To improve this, note that for each w in I , there are $\#L_w(J)$ occurrences in J . So the sum above is equal to

$$\sum_w (\#L_w(I)) \times (\#L_w(J)).$$

This equality is a restatement of the relationship between multiplication and addition(!). This second computation is much easier. First we find the frequency of k -letter words in each sequence. This is accomplished by scanning each sequence (of lengths n and m). Then the word frequencies are multiplied and added. Therefore, the total time is proportional to $4^k + n + m$. For our sequence of numbers of 2-word matches, the statistic above is

$$\begin{aligned} 0^2 + 0^2 + 0^2 + 2 \times 1 + 2 \times 1 + 2 \times 0 + 2 \times 1 + 0^2 + 0^2 + 1 \times 2 + 0 \times 1 \\ + 0^2 + 0^2 + 2 \times 1 + 0^2 + 0^2 = 10 \end{aligned}$$

If 10 is above a threshold that we specify, then a full sequence comparison can be performed. (Low thresholds require more comparisons than high thresholds.) This method is quite fast, but the comparison totally ignores the relative positions of the k -words in the sequence. A more sensitive method would be useful.

7.2.2 Binary Searches

Suppose that I and J contain the k -words listed in Table 7.1. How do we find the first word in list I, **TGAT**, within list J? In this example, we can find the matches by inspection. But what would we do if the lists were 500 entries long, and composed of words of $k = 10$? Rather than just scanning down a list from the top, a better way to find matching entries is a **binary search**. Since list J (of length m) is stored in a computer, we can extract the entry number $m/2$, which in this example is entry 16, **GACA**. Then we proceed as follows:

- Step 1: Does **TGAT** occur after entry 16 in the alphabetically sorted list? Since it occurs after entry 16, we don't need to look at the first half of the list.
- Step 2: In the second half of the list, does **TGAT** occur after the entry at position $m/2 + m/4$? This is entry 24, **TCGA**. **TGAT** occurs after this entry, so we have now eliminated the need to search in the first 3/4 of the list after only two comparisons.
- Step 3: Does **TGAT** occur after entry 24 but before entry 29? (We have split the last 1/4 of the list into two $m/8$ segments.) Since it is before 29 and after 24, we only examine the four remaining entries.
- Steps 4 and 5: Two more similar steps are needed to “zero in” on entry 25.

Had we gone through the whole list, 25 steps would have been required to find the word. With the binary search, we used only five steps. This process is analogous to finding a word in a dictionary by successively splitting the remaining pages in half until we find the page containing our word. (Try it! We found the page containing “quiescent” after ten splits of pages in a dictionary having 864 pages.)

In general, if we are searching a list of length m starting at the top and going item by item, on average we will need to search half the list before we find the matching word (if it is present). If we perform a binary search as above, we will need only $\log_2(m)$ steps in our search. This is because $m = 2^{\log_2(m)}$, and we can think of all positions in our list of length m as having been generated by $\log_2(m)$ doublings of an initial position. In the example above, $32 = 2^5$, so we should find any entry after five binary steps. In the dictionary experiment, finding the entry should have required ten steps (nine-letter word, $2^9 = 512$, and $2^{10} = 1024$ —nine “splits” are not enough since $864 > 512$).

7.2.3 Rare Words and Sequence Similarity

For the method described in Section 7.2.1, if k is large the table size can be enormous, and it will be mostly empty. For large k , another method for detecting sequence similarity is to put the k -words in an ordered list.

To find k -word matches between I and J, first break I down into a list of $n - k + 1$ k -words and J into a list of $m - k + 1$ k -words. Then put the words

Table 7.1. Ordered word lists of query sequence I and sequence J to which it is to be compared. Numbers beside each 4-word indicate the position of each word in the list. Binary searches reduce the search space by half for each iteration, checking whether the search word from I is in the remaining first or second half.

I		
TGAT		
GATG		
ATGA		
...		
J		
AAAT 1	GATG 17	
AATC 2	GATT 18	
AATG 3	GGAT 19	
ACAA 4	GGGA 20	
ATCC 5	GTCG 21	
ATGA 6	TCAC 22	
ATGT 7	TCCG 23	
ATTT 8	TCGA 24	
CAAA 9	TGAT 25	
CCGA 10	TGGG 26	
CGAA 11	TGTC 27	
CGAC 12	TTGG 28	
CGTT 13	TTTA 29	
CTTT 14	TTTC 30	
GAAT 15	TTTG 31	
GACA 16	TTTT 32	

in each list in order, from AA...A to TT...T. This takes time $n \log(n)$ and $m \log(m)$ by standard methods which are routinely available but too advanced to present here. Let's index the list by $(W(i), Pw(i))$, $i = 1, \dots, n - k + 1$ and $(V(j), Pv(j))$, $j = 1, \dots, m - k + 1$, where, for example, $W(i)$ is the i th word in the ordered list and $Pw(i)$ is the position that word had in I.

We discover k -word matches by the following algorithm which basically merges two ordered lists into one long ordered list. Start at the beginning of one list. So long as that element is smaller than the beginning of the second list continue to add elements from that list. When this is no longer the case, switch to the other list. Proceed until reaching the end of one of the lists. During this process we will discover all k -words that are equal between the lists, along with producing the merged ordered list. Because the positions in the original sequences are carried along with each k -word, we will know the location of the matches as well. Obviously matches longer than length k will be observed as successive overlapping matches.

7.3 Looking for Regions of Similarity Using FASTA

FASTA (Pearson and Lipman, 1988) is a rapid alignment approach that combines methods to reduce the search space (it depends on k -tuples) and Smith-Waterman local sequence alignment, as described in the previous chapter. As an introduction to the rationale of the FASTA method, we begin by describing **dot matrix** plots, which are a very basic and simple way of visualizing regions of sequence similarity between two different strings. We have already alluded to them in Section 5.4.

7.3.1 Dot Matrix Comparisons

Dot matrix comparisons are a special type of alignment matrix with positions i in sequence I corresponding to rows, positions j in sequence J corresponding to columns, and sequence identities indicated by placing a dot at matrix element (j, i) if the word or letter at J_j is identical to the word or letter at I_i . An example for two DNA strings is shown in Fig. 7.1A. In this example, the string CATCG in I appears twice in J, and these regions of local sequence similarity appear as two diagonal arrangements of dots: diagonals represent regions having sequence similarity.

When I and J are DNA sequences and are short, the patterns of this type are relatively easy to see. When I and J are DNA sequences and very long, there will be many dots in the matrix since, for any letter at position j in J, the probability of having a dot at any position i in I will equal the frequency of the letter J_j in the DNA. For 50% A+T, this means that on average 1/4 of the matrix elements will have a dot. When I and J are proteins, dots in the matrix elements record matches between amino acid residues at each particular pair of positions. Since there are 20 different amino acids, if the amino acid frequencies were identical, the probability of having a dot at any particular position would be 1/20.

7.3.2 FASTA: Rationale

The rationale for FASTA (Wilbur and Lipman, 1983) can be visualized by considering what happens to a dot matrix plot when we record matches of k -tuples ($k > 1$) instead of recording matches of single letters (Fig. 7.1 B and C). We again place entries in the alignment matrix, except this time we only make entries at the first position of each dinucleotide or trinucleotide (k -tuple matches having $k = 2$ (plotted numerals 2) or $k = 3$ (plotted numerals 3)). Notice how the number of matrix entries is reduced as k increases. By looking for words with $k > 1$, we find that we can ignore most of the alignment matrix since the absence of shared words means that subsequences don't match well. There is no need to examine areas of the alignment matrix where there are

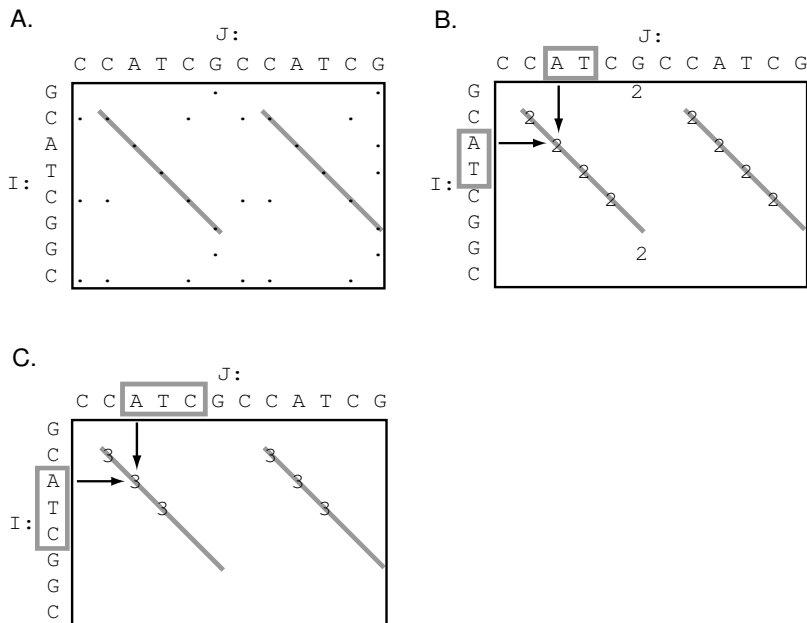


Fig. 7.1. Dot matrix plots showing regions of sequence similarity between two strings (diagonal lines). Panels A, B, and C are plots for k -tuples $k = 1, 2$, and 3 , respectively. Typical k -tuples used for plotting a particular element are indicated by boxes.

no word matches. Instead, we only need to focus on the areas around any diagonals.

Our task is now to compute efficiently diagonal sums of scores, S_l , for diagonals such as those in Fig. 7.1B. (The method for forming these scores is explained below.) Consider again the two strings I and J that we used in Section 7.2.1. There are $7 \times 11 = 77$ potential 2-matches, but in reality there are ten 2-matches with four nonzero diagonal sums. We index diagonals by the offset, $l = i - j$. In this notation, the nonzero diagonal sums are $S_{+1} = 1$, $S_0 = 4$, $S_{-5} = 1$, and $S_{-6} = 4$. It is possible to find these sums in time proportional to $n + m + \#\{k\text{-word matches}\}$. Here is how this is done.

Make a k -word list for J. In our case, this is the list for J in Table 7.2. Then initialize all row sums to 0:

$$\begin{array}{c|cccccccccccccccc}
 \ell & -10 & -9 & -8 & -7 & -6 & -5 & -4 & -3 & -2 & -1 & 0 & 1 & 2 & 3 & 4 & 5 & 6 \\
 \hline
 S_\ell & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
 \end{array}$$

Next proceed with the 2-words of I, beginning with $i = 1$, GC. Looking in the list for J, we see that $L_{GC}(J) = \{6\}$, so we know that at $l = 1 - 6 = -5$ there is a 2-word match of GC. Therefore, we replace $S_{-5} = 0$ by $S_{-5} = 0 + 1 = 1$.

Table 7.2. k -word lists for $J = \text{CCATCGCCATCG}$ and $I = \text{GCATCGGC}$, $k = 2$.

J		I	
AA		AA	
AC		AC	
AG		AG	
AT	3, 9	AT	3
CA	2, 8	CA	2
CC	1, 7	CC	
CG	5, 11	CG	5
CT		CT	
GA		GA	
GC	6	GC	1, 7
GG		GG	6
GT		GT	
TA		TA	
TC	4, 10	TC	4
TG		TG	
TT		TT	

Next, for $i = 2$, **CA**, we have $L_{\text{CA}}(J)=\{2,8\}$. Therefore replace $S_{2-2} = S_0 = 0$ by $S_0 = 0 + 1$, and replace $S_{2-8} = S_{-6} = 0$ by $S_{-6} = 0 + 1$. These operations, and the operations for all of the rest of the 2-words in I, are summarized below. Note that for each successive step, the then current score at S_i is employed: S_0 was set to 1 in step 2, so 1 is incremented by 1 in step 3.

$$\begin{array}{ll}
 i = 1, \text{ GC} & L_{\text{GC}}(J)=\{6\} \quad l = 1 - 6 = -6 \\
 & S_{-5} = 0 \rightarrow S_{-5} = 0 + 1 = 1 \\
 i = 2, \text{ CA} & L_{\text{CA}}(J)=\{2,8\} \quad l = 2 - 2 = 0 \\
 & S_0 = 0 \rightarrow S_0 = 0 + 1, \text{ and} \\
 & l = 2 - 8 = -6 \\
 & S_{-6} = 0 \rightarrow S_{-6} = 0 + 1 \\
 i = 3, \text{ AT} & L_{\text{AT}}(J)=\{3,9\} \quad l = 3 - 3 = 0 \\
 & S_0 = 1 \rightarrow S_0 = 1 + 1 = 2 \\
 & l = 3 - 9 = -6 \\
 & S_{-6} = 1 \rightarrow S_{-6} = 1 + 1 = 2 \\
 i = 4, \text{ TC} & L_{\text{TC}}(J)=\{4,10\} \quad l = 4 - 4 = 0 \\
 & S_0 = 2 \rightarrow S_0 = 2 + 1 = 3 \\
 & l = 4 - 10 = -6 \\
 & S_{-6} = 2 \rightarrow S_{-6} = 2 + 1 = 3 \\
 i = 5, \text{ CG} & L_{\text{CG}}(J)=\{5,11\} \quad l = 5 - 5 = 0 \\
 & S_0 = 3 \rightarrow S_0 = 3 + 1 = 4 \\
 & l = 5 - 11 = -6 \\
 & S_{-6} = 3 \rightarrow S_{-6} = 3 + 1 = 4
 \end{array}$$

$$\begin{aligned}
 i = 6, \text{ GG} \quad L_{\text{GG}}(\text{J}) &= \{\emptyset\} & L_{\text{GG}}(\text{J}) \text{ is the empty set: no sums} \\
 & & \text{are increased} \\
 i = 7, \text{ GC} \quad L_{\text{GC}}(\text{J}) &= \{6\} & l = 7 - 6 = 1 \\
 & & S_1 = 0 \rightarrow S_1 = 0 + 1 = 1
 \end{aligned}$$

The final result for scores of the diagonals at various offsets is

$$\begin{array}{r}
 \ell | -10 \ -9 \ -8 \ -7 \ -6 \ -5 \ -4 \ -3 \ -2 \ -1 \ 0 \ 1 \ 2 \ 3 \ 4 \ 5 \ 6 \\
 \hline
 S_\ell | 0 \ 0 \ 0 \ 0 \ 4 \ 1 \ 0 \ 0 \ 0 \ 0 \ 4 \ 1 \ 0 \ 0 \ 0 \ 0
 \end{array}$$

Notice that we only performed additions when there were 2-word matches. The algorithm that we employed is indicated in pseudocode in Computational Example 7.1.

Computational Example 7.1: Pseudocode for diagonal sums of scores

```

Set  $S_\ell \leftarrow 0$  for all  $1 - m \leq \ell \leq n - 1$ 
Compute  $L_w(\text{J})$  for all words  $w$ 
for  $i = 1$  to  $n - k - 1$ 
     $w \leftarrow I_i I_{i+1} \dots I_{i+k-1}$ 
    for  $j \in L_w(\text{J})$ 
         $\ell \leftarrow i - j$ 
         $S_\ell \leftarrow S_\ell + 1$ 
    end
end
end

```

It is possible to find local alignments using a gap length penalty of $-gx$ for a gap of length x along a diagonal. Let A_ℓ be the local alignment score and S_ℓ be the maximum of all of the A_ℓ 's on the diagonal. Then the scoring is done as follows, after initializing $A_\ell \leftarrow 0$, $S_\ell \leftarrow 0$, for each element (i, j) along the diagonal $\ell = i - j$, beginning at $i = 1$:

$$\begin{aligned}
 A_\ell &\leftarrow \max \begin{cases} A_\ell + 1 & \text{if } a_i = b_{j+l} \\ A_\ell - g & \text{if } a_i \neq b_{j+l} \\ 0 & \end{cases} \\
 S_\ell &\leftarrow \max\{S_\ell, A_\ell\}
 \end{aligned}$$

Five steps are involved in FASTA:

1. Use the look-up table to identify k -tuple identities between I and J.
2. Score diagonals containing k -tuple matches, and identify the ten best diagonals in the search space.
3. Rescore these diagonals using an appropriate scoring matrix (especially critical for proteins), and identify the subregions with the highest score (initial regions).

4. Join the initial regions with the aid of appropriate joining or gap penalties for short alignments on offset diagonals.
5. Perform dynamic programming alignment within a band surrounding the resulting alignment from step 4.

To implement the first step, we pass through I once and create a table of the positions i for each possible word of predetermined size k . Then we pass through the search space J once, and for each k -tuple starting at successive positions j , “look up” in the table the corresponding positions for that k -tuple in I. Record the i, j pairs for which matches are found. We have already illustrated this process for 2-words in Section 7.2.1. The i, j pairs define where potential diagonals can be found in the alignment matrix.

FASTA step 2 is identification of high-scoring diagonals. If I has n letters and J has m letters, there are $n + m - 1$ diagonals. (Think of starting in the lower left-hand corner, drawing successive diagonals all the way up to the top of the column representing J (m diagonals). Then continue rightward, drawing diagonals through all positions in I (n diagonals). Since you will have counted the first position twice, you need to subtract 1.) To score the diagonals, calculate the number of k -tuple matches for every diagonal having at least one k -tuple. Scoring may take into account distances between matching k -tuples along the diagonal. Note that the number of diagonals that needs to be scored will be much less than the number of all possible diagonals. Identify the significant diagonals as those having significantly more k -tuple matches than the mean number of k -tuple matches. This means, of course, that we should set a threshold, such as two standard deviations above the mean. For example, if the mean number of 6-tuples is 5 ± 1 , then with a threshold of two standard deviations, you might consider diagonals having seven or more 6-tuple matches as significant. Take the top ten significant diagonals.

In step 3, we rescore the diagonals using a scoring table and allowing identities shorter than k -tuple length. We retain the subregions with the highest scores. The need for this rescoring is illustrated by the two examples below.

I: C C A T C G C C A T C G (Number 4-tuple matches: 0)
 J: C C A A C G C A A T C A

I': C C A T C G C C A T C G
 J': A C A T C A A A T A A A

In the first case, the placement of mismatches spaced by three letters means that there are no 4-tuple matches, even though the sequences are 75% identical. The second pair shows one 4-tuple match, but the two sequences are only 33% identical. Rescoring reveals sequence similarity not detected because of the arbitrary demand for uninterrupted identities of length k .

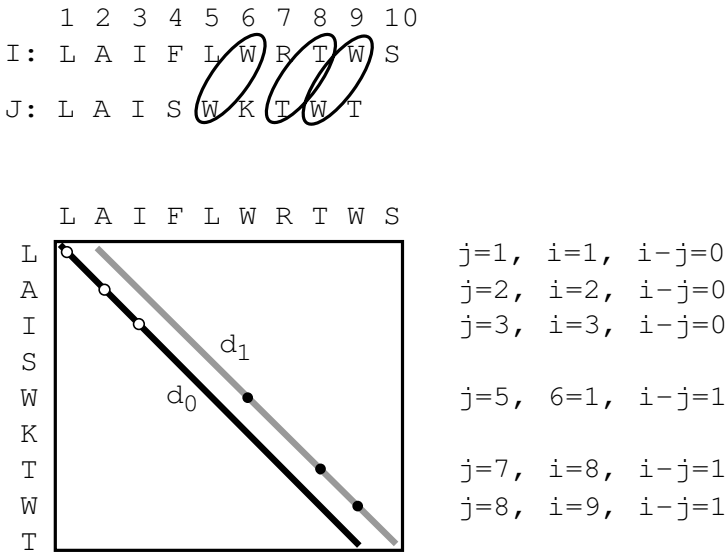
Step 4 is joining together appropriate diagonals that may be *offset* from each other, as might occur if there were a gap in the alignment (i.e., vertical or horizontal displacements in the alignment matrix, as described in the previous

chapter). Diagonal d_l is the one having k -tuple matches at positions i in string I and j in string J such that $i - j = l$. As described earlier in this chapter, $l = i - j$ is called the **offset**. Offsets are explained pictorially in Fig. 7.2. Alignments are extended by joining offset diagonals if the result is an extended aligned region having a higher alignment score, taking into account appropriate joining (gap) penalties.

Step 5 is to perform a rigorous Smith-Waterman local alignment. This can be restricted to a comparatively narrow window extending $+w$ to the right and $-w$ to the left of the positions included within the highest-scoring diagonal (see Fig. 7.3D).

7.4 BLAST

The most used database search programs are BLAST and its descendants. BLAST is modestly named for *Basic Local Alignment Search Tool*, and it was introduced in 1990 (Altschul et al., 1990). Whereas FASTA speeds up the search by filtering the k -word matches, BLAST employs a quite different



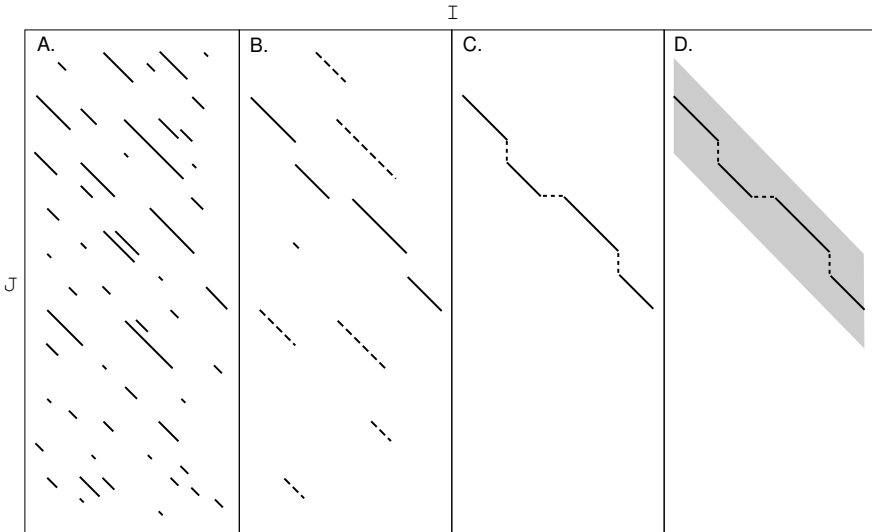


Fig. 7.3. Illustration of FASTA steps 2–5. Panel A: Identify diagonals sharing k -tuples (step 2). Panel B: Rescore to generate initial regions (step 3). Panel C: Join initial regions to give the combination having maximum score (step 4). Panel D: Perform dynamic programming in a “window space” or “band” centered around the highest-scoring initial region (step 5).

strategy. This can be summarized in two parts: the method for finding local alignments between a query sequence and a sequence in a database, and the method for producing p-values and a rank ordering of the local alignments according to their p-values. High-scoring local alignments are called “*high scoring segment pairs*,” or HSPs. The output of BLAST is a list of HSPs together with a measure of the probability that such matches would occur by chance.

7.4.1 Anatomy of BLAST: Finding Local Matches

First, the query sequence is used as a template to construct a set of subsequences of length w that can score at least T when compared with the query. A substitution matrix, containing **neighborhood sequences**, is used in the comparison. Then the database is searched for each of these neighborhood sequences. This can be done very rapidly because the search is for an exact match, just as our word processor performs exact searches. We have not developed such sophisticated tools here, but such a search can be performed in time proportional to the sum of the lengths of the sequence and the database.

Let’s return to the idea of using the query sequence to generate the neighborhood sequences. We will employ the same query sequence I and search space J that we used previously (Section 7.2.1):

$$\begin{aligned} J &= \text{C C A T C G C C A T C G} \\ I &= \text{G C A T C G G C} \end{aligned}$$

We use subsequences of length $k = 5$. For the *neighborhood size*, we use all 1-mismatch sequences, which would result from scoring matches 1, mismatches 0, and the test value (threshold) $T = 4$. For sequences of length $k = 5$ in the neighborhood of GCATC with $T = 4$ (excluding exact matches), we have:

$$\left\{ \begin{array}{c} \text{A} \\ \text{C} \\ \text{T} \end{array} \right\} \text{CATC}, \quad \text{G} \left\{ \begin{array}{c} \text{A} \\ \text{G} \\ \text{T} \end{array} \right\} \text{ATC}, \quad \text{GC} \left\{ \begin{array}{c} \text{C} \\ \text{G} \\ \text{T} \end{array} \right\} \text{TC}, \quad \text{GCA} \left\{ \begin{array}{c} \text{A} \\ \text{C} \\ \text{G} \end{array} \right\} \text{C}, \quad \text{GCAT} \left\{ \begin{array}{c} \text{A} \\ \text{G} \\ \text{T} \end{array} \right\}$$

Each of these terms represents three sequences, so that in total there are $1 + (3 \times 5) = 16$ exact matches to search for in J. For the three other 5-word patterns in I (CATCG, ATCGG, and TCGGC), there are also 16 exact 5-words, for a total of $4 \times 16 = 64$ 5-word patterns to locate in J.

A **hit** is defined as an instance in the search space (database) of a k -word match, within threshold T , of a k -word in the query sequence. There are several hits in I to sequence J. They are

5-words in I	5-words in J	J position(s)	Score
CATCG	CATCG	2, 8	5
GCATC	CCATC	1	4
ATCGG	ATCGC	3	4
TCGGC	TCGCC	4	4

In actual practice, the hits correspond to a tiny fraction of the entire search space. The next step is to extend the alignment starting from these “seed” hits. Starting from any seed hit, this extension includes successive positions, with corresponding increments to the alignment score. This is continued until the alignment score falls below the maximum score attained up to that point by a specified amount. Later, improved versions of BLAST only examine diagonals having *two* nonoverlapping hits no more than a distance A residues away from each other, and then extend the alignment along those diagonals. Unlike the earlier version of BLAST, gaps can be accommodated in the later versions.

With the original version of BLAST, over 90% of the computation time was employed in producing the ungapped extensions from the hits. This is because the initial step of identifying the seed hits was effective in making this alignment tool very fast. Later versions of BLAST require the same amount of time to find the seed hits and have reduced the time required for the ungapped extensions considerably. Even with the additional capabilities for allowing gaps in the alignment, the newer versions of BLAST run about three times faster than the original version (Altschul et al., 1997).

7.4.2 Anatomy of BLAST: Scores

The second aspect of a BLAST analysis is to rank-order the sequences found by p-values. If the database is \mathcal{D} and a sequence X scores $S(\mathcal{D}, X) = s$ against

the database, the p-value is $\mathbb{P}(S(\mathcal{D}, Y) \geq s)$, where Y is a random sequence. The smaller the p-value, the greater the “surprise” and hence the greater the belief that something real has been discovered. A p-value of 0.1 means that with a collection of query sequences picked at random, in 1/10 of the instances a score that large or larger would be discovered. A p-value of 10^{-6} means that only once in a million instances would a score of that size appear by chance alone.

There is a nice way of computing BLAST p-values that has a solid mathematical basis. Although a rigorous treatment is far beyond the scope of this book, an intuitive and accurate account is quite straightforward. In a sequence-matching problem where the score is 1 for identical letters and $-\infty$ otherwise (i.e., no mismatches and no indels), the best local alignment score is equal to the longest exact matching between the sequences. In our $n \times m$ alignment matrix, there are (approximately) $n \times m$ places to begin an alignment. Generally, an optimal alignment begins with a mismatch, and we are interested in those that extend at least t matching (identical) letters. We set

$$p = \mathbb{P}(\text{two random letters are equal}).$$

The event of a mismatch followed by t identities has probability $(1-p)p^t$. There are $n \times m$ places to begin this event, so the mean or expected number of local alignments of at least length t is $nm(1-p)p^t$. Obviously, we want this to be a rare event that is well-modeled by the Poisson distribution (see Chapter 3) with mean

$$\lambda = nm(1-p)p^t,$$

so

$$\begin{aligned} \mathbb{P}(\text{there is local alignment } t \text{ or longer}) &\approx 1 - \mathbb{P}(\text{no such event}) \\ &= 1 - e^{-\lambda} \\ &= 1 - \exp(-nm(1-p)p^t). \end{aligned}$$

This equation is of the same form used in BLAST, which estimates

$$\mathbb{P}(S(\mathcal{D}, Y) \geq s) \approx 1 - \exp(-nm\gamma\xi^t),$$

where $\gamma > 0$ and $0 < \xi < 1$. There are conditions for the validity of this formula, in which γ and ξ are estimated parameters, but this is the idea! (In BLAST output, the last quantity is called an E-value.)

The take-home message of this discussion is that the probability of finding an HSP by chance using a random query sequence Y in database \mathcal{D} is approximately equal to E .

7.5 Scoring Matrices for Protein Sequences

The alignment scores obviously depend on the scoring matrices. We discussed scoring matrices for DNA in Section 6.6. Now we seek a method to score alignments between proteins X and Y such as

$$\begin{aligned} X &= \dots NVSD\underline{V}NLNK \dots \\ Y &= \dots NASN\underline{L}SLSK \dots \end{aligned}$$

We need to assign scores for the alignment of residues at any particular position, such as the one underlined above. In other words, we need to find the probability p_{ab} of “matching” amino acid a with amino acid b . The values of the p_{ab} will differ depending on the identities of a and b . For example, the score at the position indicated in this example should take into account the hydrophobic character shared by valine and leucine, which conserves the properties (and possibly the function) of the two proteins.

7.5.1 Rationale for Scoring: Statement of the Problem

We are given two sequences, $A = a_1a_2 \dots a_n$ and $B = b_1b_2 \dots b_n$, of equal length. The alignment will be over the entire set of letters in each sequence (i.e., a global alignment). No gaps are employed in our simple illustration, although as we have seen, FASTA and BLAST do allow gaps. We seek to devise a scoring scheme based on the probabilities of matching amino acid a with amino acid b . We use an approach that we will also employ later when describing signals in DNA (Section 9.2.1). We take the ratio of two probabilities: the probability that the sequence strings match (match model \mathcal{M}) and the probability that the sequence strings were chosen at random (random model \mathcal{R}). The probability of having X and Y given the random model is

$$\mathbb{P}(A, B \mid \mathcal{R}) = \prod_i q_{a_i} \prod_i q_{b_i},$$

where q_{x_i} is the probability of occurrence of the amino acid of the type x_i in a collection of proteins, irrespective of position. The model above assumes that the identity of x_i is independent of the identity of x_{i-1} .

What is the probability of having these two sequences according to the “match” model? By “match” we recognize explicitly that amino acids at corresponding positions may have *degrees of relationship* based upon how much divergence has occurred since the two strings evolved from a common ancestor. In other words, we won’t simply be assigning a single score value for all matches and identical penalties for mismatches. We define p_{ab} as the probability of finding an amino acid of type a aligned with an amino acid of type b given that they have both evolved from an ancestor who had c at that position ($c = a, b$, or something else). This probability is

$$\mathbb{P}(A, B \mid \mathcal{M}) = \prod_i p_{a_i b_i}.$$

The score S is obtained by taking the ratio of probabilities under the two models—match relative to random sequences. This ratio is

$$\frac{\mathbb{P}(A, B \mid \mathcal{M})}{\mathbb{P}(A, B \mid \mathcal{R})} = \frac{\prod_i p_{a_i b_i}}{\prod_i q_{a_i} \prod_i q_{b_i}} = \prod_i \left(\frac{p_{a_i b_i}}{q_{a_i} q_{b_i}} \right).$$

We define the score S as

$$S = \log_2 \left(\frac{\mathbb{P}(A, B \mid \mathcal{M})}{\mathbb{P}(A, B \mid \mathcal{R})} \right) = \sum_{i=1}^n \log_2 \left(\frac{p_{a_i b_i}}{q_{a_i} q_{b_i}} \right) = \sum_{i=1}^n s(a_i, b_i),$$

which indicates that we are adding together scores for aligning individual amino acid pairs a and b :

$$s(a, b) = \log_2 \left(\frac{p_{ab}}{q_a q_b} \right).$$

The $p_{a_i b_i}$ are extracted from collections of data, as described in the next section.

7.5.2 Calculating Elements of the Substitution Matrices

What we ultimately wish to find is a **substitution matrix**, whose components are the scaled scores $s(a, b)$ for aligning amino acid a with amino acid b ,

	A	R	N	D	⋯	V
A	$s(A, A)$					
R	$s(R, A)$	$s(R, R)$				
N	$s(N, A)$	$s(N, R)$	$s(N, N)$			
D	$s(D, A)$	$s(D, R)$	$s(D, N)$	$s(D, D)$		
⋮						
V	$s(V, A)$	$s(V, R)$	$s(V, N)$	$s(V, D)$	⋯	$s(V, V)$

where $s(a, b) = s(b, a)$. (Letters labeling rows and columns are single-letter amino acid codes, listed so that the amino acid *names* are in alphabetical order: A = alanine, R = arginine, etc.)

The first substitution matrix set to be devised was the **PAM** (point accepted mutation) family (e.g., PAM100) (Dayhoff et al., 1978). With PAM100, for example, the p_{abs} used to obtain the $s(a, b)$ values are calculated so that they correspond to an average of 100 changes per 100 amino acid residues. (Note that there will still be sequence similarity after 100 changes per 100 residues since some residues will not have mutated at all, others will have changed repeatedly, and still other residues will back-mutate to their original identity.) These were evaluated by multiplying together matrices of probabilities, the originals of which depended upon a set of proteins that had diverged by a fixed amount. Now the preferred substitution matrices are the **BLOSUM** set (BLOcks SUBstitution Matrices: Henikoff and Henikoff, 1992). BLOSUM matrices are based on aligned protein sequence blocks without assumptions about mutation *rates*. BLOSUM45 is similar to PAM250. BLOSUM62 (comparable to PAM160) is commonly used (Table 7.3).

Earlier, we developed an equation for calculating the $s(a, b)$, and it requires q_a , q_b , and p_{ab} . It is obvious that the q_a (and q_b) can be obtained just by

Table 7.3. BLOSUM62 scoring matrix for scoring protein alignments. Data are in half-bits. For the meaning of single-letter IUPAC-IUB amino acid symbols, see Appendix C.1 (or <http://www.fruitfly.org/blast/blastFasta.html>). Less commonly used symbols are * = translational stop, B = D or N, and Z = E or Q. Data in this table are from <http://www.ncbi.nlm.nih.gov/Class/FieldGuide/BLOSUM62.txt>.

	A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W	Y	V	B	Z	X	*	
A	4	-1	-2	-2	0	-1	-1	0	-2	-1	-1	-1	-1	-2	-1	1	0	-3	-2	0	-2	-1	0	-4	
R	-1	5	0	-2	-3	1	0	-2	0	-3	-2	2	-1	-3	-2	-1	-1	-3	-2	-3	-1	0	-1	-4	
N	-2	0	6	1	-3	0	0	0	1	-3	-3	0	-2	-3	-2	1	0	-4	-2	-3	3	0	-1	-4	
D	-2	-2	1	6	-3	0	2	-1	-1	-3	-4	-1	-3	-3	-1	0	-1	-4	-3	-3	4	1	-1	-4	
C	0	-3	-3	-3	9	-3	-4	-3	-3	-1	-1	-3	-1	-2	-3	-1	-1	-2	-2	-1	-3	-3	-2	-4	
Q	-1	1	0	0	-3	5	2	-2	0	-3	-2	1	0	-3	-1	0	-1	-2	-1	-2	0	3	-1	-4	
E	-1	0	0	2	-4	2	5	-2	0	-3	-3	1	-2	-3	-1	0	-1	-3	-2	-2	1	4	-1	-4	
G	0	-2	0	-1	-3	-2	-2	6	-2	-4	-4	-2	-3	-3	-2	0	-2	-2	-3	-3	-1	-2	-1	-4	
H	-2	0	1	-1	-3	0	0	-2	8	-3	-3	-1	-2	-1	-2	-1	-2	-2	2	-3	0	0	-1	-4	
I	-1	-3	-3	-3	-1	-3	-3	-4	-3	4	2	-3	1	0	-3	-2	-1	-3	-1	3	-3	-3	-1	-4	
L	-1	-2	-3	-4	-1	-2	-3	-4	-3	2	4	-2	2	0	-3	-2	-1	-2	-1	1	-4	-3	-1	-4	
K	-1	2	0	-1	-3	1	1	-2	-1	-3	-2	5	-1	-3	-1	0	-1	-3	-2	-2	0	1	-1	-4	
M	-1	-1	-2	-3	-1	0	-2	-3	-2	1	2	-1	5	0	-2	-1	-1	-1	-1	1	-3	-1	-1	-4	
F	-2	-3	-3	-3	-2	-3	-3	-3	-1	0	0	-3	0	6	-4	-2	-2	1	3	-1	-3	-3	-1	-4	
P	-1	-2	-2	-1	-3	-1	-1	-2	-2	-2	-3	-3	-1	-2	-4	7	-1	-1	-4	-3	-2	-2	-1	-2	-4
S	1	-1	1	0	-1	0	0	0	-1	-2	-2	0	-1	-2	-1	4	1	-3	-2	-2	0	0	0	-4	
T	0	-1	0	-1	-1	-1	-1	-2	-2	-1	-1	-1	-1	-2	-1	1	5	-2	-2	0	-1	-1	0	-4	
W	-3	-3	-4	-4	-2	-2	-3	-2	-2	-3	-2	-3	-1	1	-4	-3	-2	11	2	-3	-4	-3	-2	-4	
Y	-2	-2	-2	-3	-2	-1	-2	-3	2	-1	-1	-2	-1	3	-3	-2	-2	2	7	-1	-3	-2	-1	-4	
V	0	-3	-3	-3	-1	-2	-2	-3	-3	3	1	-2	1	-1	-2	-2	0	-3	-1	4	-3	-2	-1	-4	
B	-2	-1	3	4	-3	0	1	-1	0	-3	-4	0	-3	-3	-2	0	-1	-4	-3	-3	4	1	-1	-4	
Z	-1	0	0	1	-3	3	4	-2	0	-3	-3	1	-1	-3	-1	0	-1	-3	-2	-2	1	4	-1	-4	
X	0	-1	-1	-1	-2	-1	-1	-1	-1	-1	-1	-1	-1	-1	-2	0	0	-2	-1	-1	-1	-1	-1	-4	
*	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	1	

counting the number of occurrences of each amino acid type in an appropriate collection of protein sequences, and then dividing by the total number of amino acids represented. But we are still left with the problem of where to obtain the p_{ab} . There are a number of repositories of protein data that are extremely useful for obtaining both types of quantities:

SWISS-PROT (<http://au.expasy.org/sprot/>)

This is an annotated database of protein sequences, that is minimally redundant (multiple entries for the same sequence are avoided) and heavily cross-indexed with other protein databases. At this time, there are about 154,000 protein sequences representing 57×10^6 letters in this database.

PROSITE (<http://www.expasy.ch/prosite/>)

This is a database of protein families and signature motifs (characteristic short sequence patterns) that characterize these families. Approximately 1700 families and domains are archived.

BLOCKS (<http://blocks.fhcrc.org/>)

This is a compilation of the most highly conserved regions for proteins in PROSITE. Here are listed multiply aligned, ungapped, conserved segments characteristic of each protein family. Examples are shown in Fig. 7.4.

Block PR00851A

```
ID XRODRMPGMNTB; BLOCK
AC PR00851A; distance from previous block=(52,131)
DE Xeroderma pigmentosum group B protein signature
BL adapted; width=21; seqs=8; 99.5%=985; strength=1287
XPB_HUMAN|P19447 ( 74) RPLWVAPDGHIFLEAFSPVYK 54
XPB_MOUSE|P49135 ( 74) RPLWVAPDGHIFLEAFSPVYK 54
P91579 ( 80) RPLYAPDGHIFLESFSPVYK 67
XPB_DROME|Q02870 ( 84) RPLWVAPNGHVLFLESFSPVYK 79
RA25_YEAST|Q00578 (131) PLWISPSDGRIILESFSPVYK 100
Q38861 ( 52) RPLWACADGRIFLETFSPLYK 71
O13768 ( 90) PLWINPIDGRIILESFSPVYK 100
O00835 ( 79) RPIWVCPDGHIFLETFSAIYK 86
//
```

Block PR00851B

```
ID XRODRMPGMNTB; BLOCK
AC PR00851B; distance from previous block=(65,65)
DE Xeroderma pigmentosum group B protein signature
BL adapted; width=20; seqs=8; 99.5%=902; strength=1435
XPB_HUMAN|P19447 ( 160) TVSYGKVKLVKLNRYFVES 68
XPB_MOUSE|P49135 ( 160) TVSYGKVKLVKLNRYFVES 68
P91579 ( 166) TQSYGKVKLVKLNRYFVES 85
XPB_DROME|Q02870 ( 170) TLSYGKVKLVKLNRYFVES 77
RA25_YEAST|Q00578 (217) TISYGKVKLVKLNRYFVES 100
Q38861 ( 138) TANYGKVKLVKLNRYFVES 90
O13768 ( 176) TVSYGKVKLVKLNRYFVES 72
O00835 ( 165) TQSYGKVKLVKLNRYFVES 87
//
```

Fig. 7.4. Examples of sequence blocks from the Blocks database. In this case, two different blocks from the same set of proteins are presented. All proteins are related to a human gene associated with the DNA-repair defect xeroderma pigmentosum (leading to excessive sensitivity to ultraviolet light). Reprinted, with permission, from the Blocks database (<http://blocks.fhcrc.org/>). Copyright 2003 Fred Hutchinson Cancer Research Center. Data for entry PR00851 adapted from the Prints database. (<http://www.bioinf.man.ac.uk/dbrowser/PRINTS>).

7.5.3 How Do We Create the BLOSUM Matrices?

Mechanics for enumerating the occurrences of various types of paired amino acids (as a preliminary to calculation of p_{ab}) are as follows. Each block consists of n aligned sequences each having w residues. For each column in a block (archived in the Blocks database), we count the number of pairwise matches and mismatches for each amino acid type. In column 3 of the block shown below (underlined residues) we find that the number of pairwise matches of L with L is $4 + 3 + 2 + 1 = 10$ or $5(5 - 1)/2$. (Matches of L in a sequence to itself are not counted.)

```

R P L W V A P D ...
R P L W V A P D ...
R P L Y L A P D ...
R P L W V A P N ...
P L W I S P S D ...
R P L W A C A D ...
P L W I N P I D ...
R P I W V C P D ...

```

The enumeration of matches between all leucine (L) residues in *the same column* can be understood by the matrix below, which applies to the indicated column:

	L	L	L	L	W	L	W	I
L	-	+	+	+		+		
L		-	+	+			+	
L			-	+				+
W					-			
L						-		
W							-	
I								-

From this we see that the total number of pairwise matches per column is equal to the number of “+” entries in the triangular area above the diagonal. If we take the total number of entries in the matrix (n^2), subtract the number of entries on the diagonal (n), and divide by 2 (to avoid counting the match of L_1 with L_2 as both L_1L_2 and L_2L_1), we obtain a total of $n(n - 1)/2$ possible pairings. Each block then provides $w \times n(n - 1)/2$ possible pairings, and we count the number of pairings of each type for more than 24,000 blocks. We keep a running total of the number of each kind of pairing.

Now make a matrix for the number of occurrences f_{ab} for each pairing of each type:

$$\begin{array}{cccccc}
& \text{A} & \text{R} & \text{N} & \text{D} & \cdots & \text{V} \\
\text{A} & f_{\text{A,A}} & & & & & \\
\text{R} & f_{\text{R,A}} & f_{\text{R,R}} & & & & \\
\text{N} & f_{\text{N,A}} & f_{\text{N,R}} & f_{\text{N,N}} & & & \\
\text{D} & f_{\text{D,A}} & f_{\text{D,R}} & f_{\text{D,N}} & f_{\text{D,D}} & & \\
& \vdots & & & & & \\
\text{V} & f_{\text{V,A}} & f_{\text{V,R}} & f_{\text{V,N}} & f_{\text{V,D}} & \cdots & f_{\text{V,V}}
\end{array}$$

We can use these f_{ab} to calculate p_{ab} using the following equation:

$$p_{ab} = f_{ab} / \sum_{a=1}^{20} \sum_{b=1}^a f_{ab}.$$

Notice the limits on the summation in the denominator. Think of the first summation as extending over rows and the second over columns. Since $p_{ab} = p_{ba}$, we are interested in the diagonal together with the terms below it. Therefore, the second summation (performed for each row a) needs only to proceed up to column a , as indicated. The denominator is the total number of pairwise matches, including residue a with itself.

Using the same reasoning as above, the number of terms below the diagonal is $20(20 - 1)/2 = 190$. The number of terms on the diagonal is 20, so there are 210 distinct p_{abs} . We get the estimated probability for each amino acid, q_a , based on its frequency of occurrence in the whole collection of blocks:

$$q_a = \sum_{b=1}^{20} \left(f_{ab} / \sum_{a=1}^{20} \sum_{b=1}^a f_{ab} \right).$$

Then we take $s(a, b) = \log_2(p_{ab}/q_a q_b)$. In practice, these values are rounded off and scaled. So, for BLOSUM, the scores are reported in half-bits: $s(a, b)$ [half-bits] = $2s(a, b)$.

There is one more matter to deal with: How do we “tune” the matrix to the amount of sequence divergence we are expecting in our similarity search? This has to do with how we count pairs in blocks. Consider Fig. 7.5, where the tips of the dendrogram indicate the identity of the amino acid at a particular position in members of a block of eight sequences.

Obviously, there is a group of four sequences that are very similar to each other that have L at that position. Should we count the top four examples as separate individuals for evaluating the f_{ij} ? Normally, clustering is performed for the sequence entries in the blocks (clustering will be discussed in Chapter 10), and contributions from sequences that cluster at similarities greater than some specified cutoff (broken line) are averaged. In this case (where all letters are identical), the effect would be that of replacing four Ls with one L prior to counting the number of pairwise matches. Moving the cutoff to lower levels of similarity produces a BLOSUM matrix whose entries correspond to greater amounts of evolutionary separation.

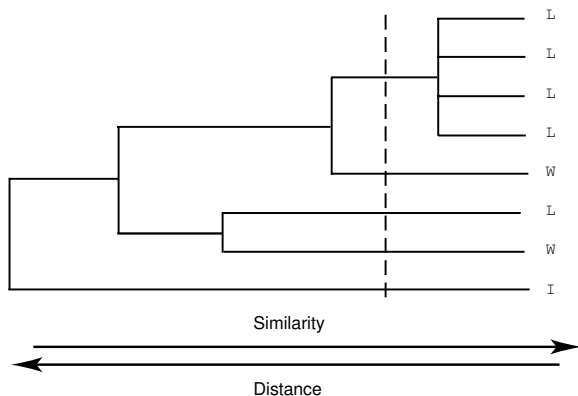


Fig. 7.5. Calculating f_{ab} as a function of evolutionary distance. The dendrogram (branching pattern) is based upon the evolutionary distance between the proteins from which the sequences in a particular block were taken. The single-letter amino acid codes on the right represent those amino acids present at a particular position in the sequences constituting the sequence block. If we are concerned with recently diverged proteins, each of the four L residues in the cluster at the top should be counted separately. If the concern is with more distantly related proteins (with distance indicated by the dotted cutoff line), then the cluster of four L residues should only be counted as one instance of L instead of four.

Computational Example 7.2: Using BLOSUM matrices

Score the alignment

```

M Q L E A N A D T S V
  :   :   :
L Q E Q A E A Q G E M

```

Using the BLOSUM62 scoring matrix (Table 7.3), we see that

$$\begin{aligned}
 S &= 2 + 5 - 3 - 4 + 4 + 0 + 4 + 0 - 2 + 0 + 1 \\
 &= 7 \text{ (half-bits)}.
 \end{aligned}$$

Now that we have seen how to obtain and use BLOSUM matrices, we should examine Table 7.3 to make sure that the scores make biological sense. The largest score (11 half-bits) is for conservation of tryptophan (W) in two sequences. Tryptophan is a relatively rare amino acid, so there should be a larger “reward” when it appears at corresponding positions in an alignment of two sequences. The lowest scores are -4 for aligning a translational stop * with any other amino acid or some unfavorable alignments such as D opposite L. The low score in the latter case can be understood because aspartic acid

(D) codons GAC and GAU are at least two mutations away from those of leucine (L)(UUA, UUG, CUA, CUC, CUG, CUU) and because the properties of D (polar and negatively charged) are quite different from those of L (nonpolar and neutral). In contrast, aligning isoleucine (I) opposite leucine produces a positive score of 2 half-bits. All three codons for isoleucine (AUA, AUC, AUU) are only one mutation away from a leucine codon, and isoleucine has the same properties as leucine (nonpolar and neutral). Thus it is relatively easier to produce this mutation, and the mutation is more likely to be tolerated under selection. In summary, if the alignment is between chemically similar amino acids, the score will be positive. It will be zero or negative for dissimilar amino acid residues. Also, when aligning an amino acid with itself, scores for aligning rare amino acids are larger than scores for aligning common ones.

7.6 Tests of Alignment Methods

At this point, we should remind ourselves *why* we are performing alignments in the first place. In many cases, the purpose is to identify homologs of the query sequence so that we can attribute to the query annotations associated with its homologs in the database. The question is, “What are the chances of finding in a database search HSPs that are *not* homologs?” Over evolutionary time, it is possible for sequences of homologous proteins to diverge significantly. This means that to test alignment programs, some approach other than alignment scores is needed to find homologs. Often the three dimensional structures of homologs and their domain structures will be conserved, even though the proteins may have diverged in sequence. Structure can be used as a criterion for identifying homologs in a test set.

A “good” alignment program meets at least two criteria: it maximizes the number of homologs found (true positives), and it minimizes the number of nonhomologous proteins found (false positives). Another way to describe these criteria is in terms of *sensitivity* and *specificity*, which are discussed in more detail in Chapter 9. In this context, sensitivity is a measure of the fraction of actual homologs that are identified by the alignment program, and the specificity is a measure of the fraction of HSPs that are not actually homologs. Brenner et al. (1998) tested a number of different alignment approaches, including Smith-Waterman, FASTA, and an early version of BLAST. They discovered that, at best, only about 35% of homologs were detectable at an error frequency of 0.1% per query sequence.

An intuitive measure of homology employed in the past was the percentage of sequence identity. The rule of thumb was that sequence identities of 25%–30% in an alignment signified true homology. Brenner et al. employed a database of known proteins annotated with respect to homology/non-homology relationships. Their results are shown in Fig. 7.6. Figure 7.6B shows percentage identity plotted against alignment length for proteins that are *not* homologs. For comparison, a threshold percentage identity taken to imply

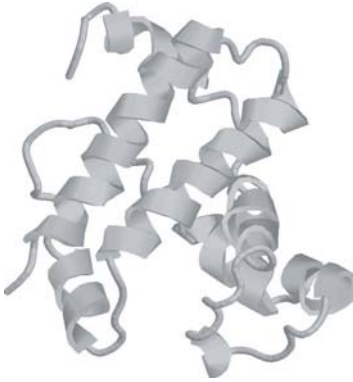
similar structure is plotted as a line (see Brenner et al., 1998 for details). The point is that for alignments 100 residues in length, about half of the *nonhomologous* proteins show *more* than 25% sequence identity. At 50 ± 10 residues of alignment length, there are a few nonhomologous proteins having over 40% sequence identity. A particular example of this is shown in Fig. 7.6A. This serves as a reminder of why methods providing detailed statistical analysis of HSPs are required (Section 7.4.2).

References

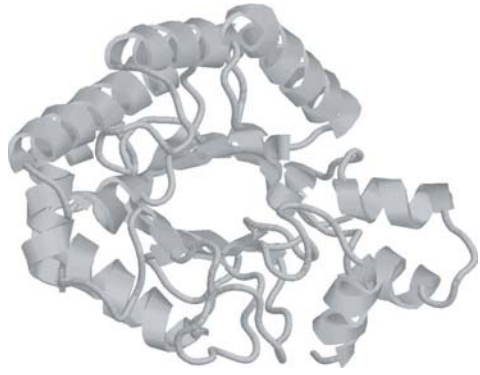
- Altschul SF, Gish W, Miller W, Myers EW, Lipman DJ (1990) Basic local alignment search tool. *Journal of Molecular Biology* 215:403–410.
- Altschul SF, Madden TL, Schaffer AA, Zhang J, Zheng Z, Miller W, Lipman DJ (1997) Gapped BLAST and PSI-BLAST: A new generation of protein database search programs. *Nucleic Acids Research* 25:3389–3402.
- Brenner SE, Chothia C, Hubbard TJP (1998) Assessing sequence comparison methods with reliable structurally identified distant evolutionary relationships. *Proceedings of the National Academy of Sciences USA* 95:6073–6078.
- Dayhoff MO, Schwartz RM, Orcutt BC (1978) A model of evolutionary change in proteins. In Dayhoff MO (ed) *Atlas of Protein Sequence and Structure*, Vol. 5, Suppl. 3. Washington D.C.:National Biomedical Research Foundation, pp. 345–352.
- Henikoff S, Henikoff JG (1992) Amino acid substitution matrices from protein blocks. *Proceedings of the National Academy of Sciences USA* 89:10915–10919.
- Pearson WR, Lipman DJ (1988) Improved tools for biological sequence comparison. *Proceedings of the National Academy of Sciences USA* 85:2444–2448.
- Wilbur WJ, Lipman DJ (1983) Rapid similarity searches of nucleic acid and protein data banks. *Proceedings of the National Academy of Sciences USA* 80:726–730.

Fig. 7.6 (opposite page). The limitations of sequence identity as an indicator of homology. Panel A: Unrelated proteins that have a 40% sequence identity over a segment of approximately 60 residues. Panel B: Scores of unrelated, nonhomologous proteins as a function of alignment length. The line indicates the sequence identity cutoff, sometimes taken as an indicator of homology. Reprinted, with permission, from Brenner SE et al. (1998) *Proceedings of the National Academy of Sciences USA* 95:6073–6078. Copyright 1998 National Academy of Sciences, USA.

A.



Hemoglobin β -chain (1hdsb)

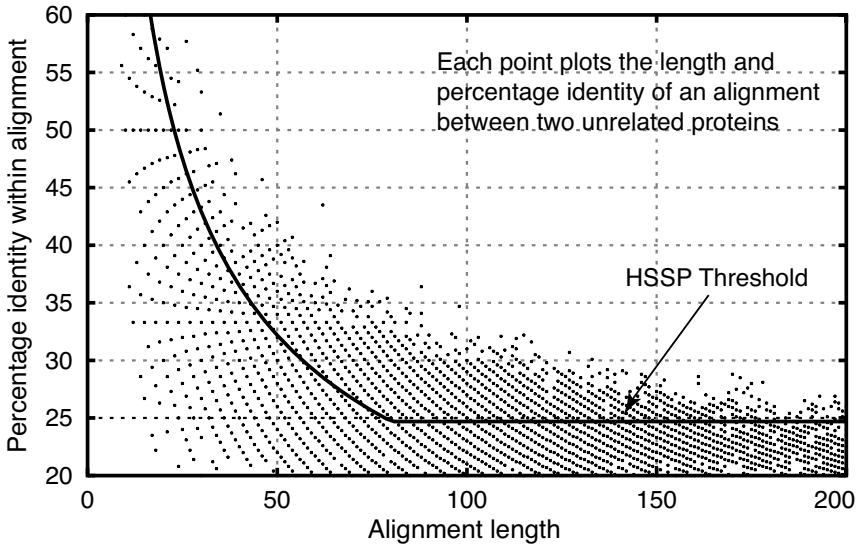


Cellulase E2 (1tml_)

```

1hdsb GKVDVDVVGAQALGR--LLVVYPWTQRFFQHFGNLSSAGAVMNNPKVKAHGKRVLDAFTQGLKH
      .....:.....:.....:.....:.....:.....:.....:.....:.....:.....:.....
1tml_ GQVDALMSAAQAAGKIPIILVVYNAPGR--DCGNHSSGGA---PSHSAY-RSWIDEFAAGLKN
    
```

B.



Exercises

Exercise 1. Find “serendipity” in a dictionary using splits as described in the text. How many splits were required? Compare this number with $\lceil \log_2(\# \text{ pages}) \rceil$. Suppose that each split divided the pages remaining into thirds instead of halves. What is the formula relating the number of steps to the number of pages in that case?

Exercise 2. In Section 7.2.1, it was suggested that the statistic

$$\sum_w (\#L_w(I)) \times (\#L_w(J))$$

could be used to determine quickly whether I and J share sufficient sequence similarity to justify a “full-bore” dynamic programming alignment. For DNA with $k = 3$, $\text{length}(I) = 100$, $\text{length}(J) = 1000$, indicate how you could use simulation with iid letters to set a threshold value for this statistic, for deciding when to employ dynamic programming.

Exercise 3. Suggest an alternative to FASTA for rapidly searching a large search space $J \gg I$ using the approach in Section 7.2.1. Would you need to modify your method for setting the threshold?

Exercise 4. For the sequences $I = \text{GCATCGGC}$ and $J = \text{CCATCGCCATCG}$, find matching 4-words shared by I and J, as described in Section 7.2.3. Do this by making a table similar to Table 7.2, but only listing 4-words that actually occur in I or J. (Otherwise, the table would have $4^4 = 256$ rows!)

Exercise 5. Compute the average number of non-empty elements in a dot matrix comparison for $k = 1$, with I and J both drawn from a human DNA sequence (41% G+C).

Exercise 6. Given strings X and Y, each having differing base compositions, write out the formula for calculating $p = \mathbb{P}(\text{two random letters are equal})$, defined in Section 7.4.2. Clearly define the symbols that you use.

Exercise 7. For $I = \text{TTGGAATACCATC}$ and $J = \text{GGCATAATGCACCCC}$, make dot matrices for the k -tuple hits for $k = 1, 2$, and 3.

Exercise 8. The *E. coli* F plasmid transfer origin region contains the sequence

I:
 5'-ATAAATAGAGTGTATGAAAAATTAGTTTCTTACTCTCTTTATGATATTT
 AAAAAAGCG-3'

The TraY protein has been shown to bind to a region some 1600 bp away at a site that contains the sequence

J:
 5'-TAACACCTCCCGCTGTTTAT-3'

Perform dot-matrix analyses for $k = 1$ and $k = 2$ to locate in I a subsequence that is similar to J. This identifies a potential binding site for TraY in sequence I. [Hint: Use R to construct a matrix having dimensions determined by the lengths of J and I, initialize elements to zero, and then substitute the appropriate elements with the number 1 to indicate matches. Produce two matrices: one for $k = 1$ and another for $k = 2$.]

Exercise 9. For I = GTATCGGCGC and J = CGGTTCGTATCGTCG, make a 2-word list for J. Then execute the FASTA algorithm (Example 7.1) beginning with $S_i = 0$. Compute S_i as you go through I, beginning at $i = 1$ and the 2-word GT, and ending at $i = 9$.

Exercise 10. To search J = CGGTTCGTATCGTCG for matches to TTCG within one mismatch, first make a list of all possible matches. How many matches are there within a single mismatch neighborhood of TTCG? [Hint: There is one exact pattern, and there are $3 \times 4 = 12$ single-mismatch patterns.]

Exercise 11. Download AB125166 and X68309 from the NCBI nucleotide databases (see Appendix B for the URL). Edit both sequences so that they contain the first 1000 positions in FASTA format. Then perform a Smith-Waterman local alignment using resources at <http://www.cmb.usc.edu/>, setting mismatch and gap parameters at 1000, and requesting return of the top 100 alignments.

- How many alignments are there of length $t \geq 8$?
- Use the expression for λ in Section 7.4.2 to compute the expected number of alignments of length at least 8 for sequences of this size (see Exercise 4 for computing p).
- Use R to simulate ten pairs I and J of iid sequences having the same base compositions as in the first 1000 nucleotides of AB125166 and X68309. Then perform the Smith-Waterman alignment on each of the 10 pairs, and calculate the average number of alignments of length at least 8. Compare your result with those from part a and part b above, and explain agreements or disagreements.
- The probability that there is a local alignment of length t or more is approximately

$$1 - \exp[-nm(1 - p)^t].$$

Calculate the probability (called a p-value) for $t =$ optimal alignment score in part a. What do you conclude from this p-value? Explain your answer carefully.

Exercise 12. Using only the data contained in the two blocks shown in Fig. 7.4, compute f_{RR} and p_{RR} as defined in Section 7.5.3. [Hint: Only columns that contain two or more instances of an R residue need be considered.]

Note: Because the computation above uses only a tiny sample of the total number of blocks available, the result computed here is not expected to lead to a score that agrees with the one in a BLOSUM matrix.

Exercise 13. To test the probability of irrelevant hits from a BLAST search, download the first two paragraphs of Jane Austen's *Emma* from a Web page located with a search engine of your choice. Remove all spaces and punctuation, and then replace letters "o" and "u" by "a" (alanine) and letters "b", "x", "j", and "z" by "g" (glycine). This should yield a string having about 500 characters. Add a first line `>emma` to convert to FASTA format, and then run WU-BLAST2 for proteins on the server at the European Bioinformatics Institute (see Appendix B). What are the E values and percentage identities for the top three sequences? How do these compare with real biological sequences?

DNA Sequence Assembly

8.1 The Biological Problem

The ultimate physical map of a genome is its sequence. For free-living organisms, genomes may be represented by strings of 5×10^5 to 10^{10} characters, depending upon the genome size of the organism. However, current sequencing technologies allow accurate reading of no more than 500 to 800 bp of contiguous DNA sequence. This means that the sequence of an entire genome must be assembled from collections of comparatively short subsequences. This process is called DNA sequence **assembly**. These sequences are often, but not always, generated from relatively short (~ 2 kb) inserts contained in M13 or small plasmid vectors. Obviously, experimental strategies are needed to convert a collection of sequences derived from large numbers of small inserts to the complete genome sequence. There have been two general approaches.

One approach, which was assumed to be optimal at the start of the Human Genome Project, is called the **top-down** (or map-based) approach. In its strictest form, the idea is to clone the genome into a hierarchy of libraries having successively smaller inserts (e.g., making a YAC library having a low-resolution restriction map, cosmid libraries from each YAC with a higher-resolution restriction map, and small plasmid libraries from each cosmid). The location of each clone relative to the restriction map of the larger-insert clone from which it was derived is tracked. At the end, the small-insert clone can be sequenced, and the map position of the sequence can be obtained by tracing back up the hierarchy of mapped inserts. In this case, much of the sequence assembly problem is solved by the prior restriction mapping.

Another approach to mapping is the **bottom-up** approach, in which a small-insert library is used to assemble the restriction map of a larger region by detecting overlaps of inserts to build contigs. We have already discussed this approach to restriction mapping in Chapter 4. But suppose that, instead of having a restriction map of the cloned inserts, we knew their DNA sequences. Just as with contig assembly from the restriction mapping data, we can detect overlaps with sequences in other inserts and assemble a *sequence* rather than a

restriction map. It is this process that we will be discussing in this chapter. The process of producing the sequence of a DNA segment (perhaps a genome) from a large number of randomly chosen sequence reads derived from it is called **shotgun sequencing**. **Whole-genome shotgun (WGS)** sequencing has produced a rapid expansion in our knowledge of genomes. As we will see at the end of this chapter, hybrid strategies combining elements of both the top-down and bottom-up approaches are often useful in actual practice.

8.2 Reading DNA

For many years, determining DNA sequence was extremely difficult. The first rapid sequencing method was developed by Maxam and Gilbert (1977). This approach is a traditional analytical one in the sense that it depends upon chemically breaking down a more complicated molecule into smaller pieces, which are then analyzed. Though revolutionary for its time, this method has now been largely superseded by the Sanger **dideoxy sequencing** method (Sanger et al., 1977), which employs the counterintuitive approach of analysis by synthesis. The Sanger approach is easier to automate and is the basis for high-throughput, high-volume sequencing “factories.”

This method depends upon the properties of DNA polymerases and the biochemistry of DNA synthesis. Rather than breaking down a duplex molecule into fragments to be sequenced, we start with a *single-strand* molecule and manufacture fragments whose sizes depend upon the actual DNA sequence. We need to recall the biochemical requirements for DNA synthesis *in vitro*: a DNA template, a primer, four deoxynucleoside triphosphates (precursors of the DNA to be synthesized—dATP, dCTP, dGTP, and dTTP), and DNA polymerase in appropriate buffers.

8.2.1 Biochemical Preliminaries

The **template** is the DNA sequence to be copied (in this case, sequenced). This is supplied as a cloned insert in a small plasmid or bacteriophage. Bacteriophage M13 was originally used as the cloning vector for sequencing because it produced single-stranded circular molecules that could be used directly as templates. Later, M13 replication origins were incorporated into small plasmid vectors (yielding vectors called “phagemids”). Single-strand phagemid DNA circles could then be produced from a bacterial growth culture by adding an M13 *helper phage* to provide the necessary phage replication genes. Sanger dideoxy sequencing can also be performed on duplex plasmid or phagemid substrates by denaturing the DNA. It is not necessary to remove the opposite (nontemplate) strand.

DNA polymerases are able to synthesize DNA by adding deoxynucleoside triphosphates to a pre-existing 3'-OH on a DNA strand that is base-paired to a template. Such a short DNA strand (RNA *in vivo*) is called a **primer**. The

Sanger method employs synthetic primers that are complementary to DNA immediately adjacent to the cloned insert. For some vectors, these have been called *universal sequencing* primers. Since the primers are complementary to vector sequences—not to insert sequences—the same primer can be employed for different clones (i.e., custom primer synthesis for each clone is not needed). Different primers are available for the two sides of the cloning site into which the insert is placed. This allows sequencing from either end of the insert (i.e., from opposite strands of the insert since DNA is antiparallel). For reasons we explain later, it is often useful to track which strand (i.e., which primer) was used for every read of sequence generated.

Deoxynucleoside triphosphates are the precursors for DNA synthesis. Remember that “deoxy” means that, unlike RNA precursors, DNA precursors lack an $-OH$ group at the $2'$ position of the ribose (sugar) portion of the molecule. During DNA synthesis, the $5'$ end of the dXTP (dXTP means *any* deoxynucleoside triphosphate) is joined to the $3'$ end of the growing chain, with the elimination of pyrophosphate, yielding a product chain extended by one more residue, whose new $3'$ end is available for the next addition. The identity of the incorporated dXTP is determined by the template and the Watson-Crick base-pairing rules: if the template contains a T residue, then dATP is used by the polymerase to extend the chain; if the template contains a C residue, then dGTP is used to extend the chain, and so on.

DNA polymerases are the enzymes that can catalyze template-directed DNA synthesis. Other enzymes, such as AMV reverse transcriptase, may sometimes be employed as well. Many polymerases (*E. coli* DNA polymerase I being a classic example) contain *exonuclease* activities: they are capable also of digesting DNA ahead of the growing chain, or going back and digesting the growing chain itself. These capabilities are called, respectively, $5' \rightarrow 3'$ exonuclease and $3' \rightarrow 5'$ exonuclease activities. Because these are irrelevant for DNA sequencing, commercial sequencing polymerases have been genetically engineered to remove these activities if present. One commercial sequencing polymerase is Sequenase[®], which is an engineered version of phage T7 DNA polymerase. It lacks the normal $3' \rightarrow 5'$ exonuclease activity. Sequencing polymerases are selected or engineered to be highly *processive* (adding many residues to the growing chain without falling off the template), to be relatively insensitive to template secondary structure (to prevent “stalling”), and to be able to efficiently use nonconventional substrates when required (e.g., thio-dXTPs or dideoxynucleoside triphosphates—ddXTPs). Other thermostable polymerases derived from *Thermus aquaticus* (Taq polymerases) are used in cycle sequencing protocols, which require periods at elevated temperatures.

This may seem like a lot of “fussy” biochemical detail, but the quality of the sequence depends upon things such as the particular template and polymerase that are employed. Unresolved problems for any particular sequencing run may require computational methods for “teasing out” the actual sequence.

Proper attention to biochemical matters will maximize the length of unambiguous sequence produced from each sequencing reaction.

8.2.2 Dideoxy Sequencing

Sanger dideoxy sequencing employs chain-terminating dideoxynucleoside triphosphates, ddXTPs, to produce DNA molecules extending from the primer into the template and stopping at positions specific to the ddXTP and to the type of template residue with which it base-pairs. The reaction products are then resolved on polyacrylamide gels for size analysis. The method is illustrated in Fig. 8.1. In this illustration, we assume that an appropriate fluorescent or radioactive label has been applied to the primer and that four separate reactions are performed, one to identify positions of each of the four bases in the DNA. We indicate later other formats for setting up the reactions.

In an ordinary polymerization reaction containing all four dXTPs and *no* ddXTPs, polymerization continues from the primer sequence all the way to the 5' end of the template (if the enzyme is processive enough and there are enough reagents in the reaction mixture) (Fig. 8.1B). With dideoxy sequencing, each reaction solution is “doped” with a small amount of an appropriate dideoxynucleoside triphosphate. For the “A” reaction shown in the figure, a small amount of ddATP is included in addition to dATP. As polymerization proceeds along the many templates in the reaction solution, most of the time the polymerases will pick up a dATP from solution and insert A opposite T, leaving a 3'-OH to allow continued growth of the chain. Occasionally, however, the polymerases will insert a ddATP opposite T, and when this happens, there is *no* 3'-OH, and that particular chain can no longer grow—it is terminated (Fig. 8.1C). A termination can, in principle, occur opposite any T in the template, and different chain lengths are produced in terminations that have occurred in different positions. Because there are so many templates in the reaction solution, there will be many terminated chains for each possible position.

Figure 8.1D shows the types of terminated molecules sorted by size from largest to smallest. Polyacrylamide gel electrophoresis *physically* sorts molecules by size, with the largest near the origin (top) of the gel and the

Fig. 8.1 (opposite page). Principles of Sanger dideoxy chain-termination sequencing. The primer sequence and its complement are enclosed in the grey boxes. Results for the “A” sequencing reaction are illustrated. A and T are in uppercase boldface type, and bases that are not involved in this specific chain-termination reaction are all in lowercase type. Panel A: Template DNA with hybridized primer: substrate for copying by DNA polymerase. Panel B: Product of DNA polymerization if no dideoxynucleoside triphosphates are added. Panel C: Different products produced after DNA synthesis, with termination occurring at various locations as a result of incorporation of ddATP. The * indicates a dideoxy residue. Panel D: Newly synthesized products sorted in order of decreasing size.

A. 5' **catgacgatcgg**3'
3' **gtactgctagcc**aaa**TggacaaTagcT**acagaccca**TTTcTgaT**cagg...5'

B. 5' **catgacgatcgg**ttt**AcctgttAtcgAtgtctgggtAAA**gAct**Ag**tcc...3'
3' **gtactgctagcc**aaa**TggacaaTagcT**acagaccca**TTTcTgaT**cagg...5'

C. 5' **catgacgatcgg**ttt**A***
3' **gtactgctagcc**aaa**TggacaaTagcT**acagaccca**TTTcTgaT**cagg...5'

5' **catgacgatcgg**ttt**AcctgttAtcgAtgtctgggtAA***
3' **gtactgctagcc**aaa**TggacaaTagcT**acagaccca**TTTcTgaT**cagg...5'

5' **catgacgatcgg**ttt**AcctgttA***
3' **gtactgctagcc**aaa**TggacaaTagcT**acagaccca**TTTcTgaT**cagg...5'

5' **catgacgatcgg**ttt**AcctgttAtcgAtgtctgggtAAA**gAct**A***
3' **gtactgctagcc**aaa**TggacaaTagcT**acagaccca**TTTcTgaT**cagg...5'

5' **catgacgatcgg**ttt**AcctgttAtcgAtgtctgggtAAA**g**A***
3' **gtactgctagcc**aaa**TggacaaTagcT**acagaccca**TTTcTgaT**cagg...5'

5' **catgacgatcgg**ttt**AcctgttAtcgA***
3' **gtactgctagcc**aaa**TggacaaTagcT**acagaccca**TTTcTgaT**cagg...5'

5' **catgacgatcgg**ttt**AcctgttAtcgAtgtctgggtA***
3' **gtactgctagcc**aaa**TggacaaTagcT**acagaccca**TTTcTgaT**cagg...5'

5' **catgacgatcgg**ttt**AcctgttAtcgAtgtctgggtAAA***
3' **gtactgctagcc**aaa**TggacaaTagcT**acagaccca**TTTcTgaT**cagg...5'

D. 5' **catgacgatcgg**ttt**AcctgttAtcgAtgtctgggtAAA**gAct**A***

5' **catgacgatcgg**ttt**AcctgttAtcgAtgtctgggtAAA**g**A***

5' **catgacgatcgg**ttt**AcctgttAtcgAtgtctgggtAAA***

5' **catgacgatcgg**ttt**AcctgttAtcgAtgtctgggtAA***

5' **catgacgatcgg**ttt**AcctgttAtcgAtgtctgggtA***

5' **catgacgatcgg**ttt**AcctgttAtcgA***

5' **catgacgatcgg**ttt**AcctgttA***

5' **catgacgatcgg**ttt**A***

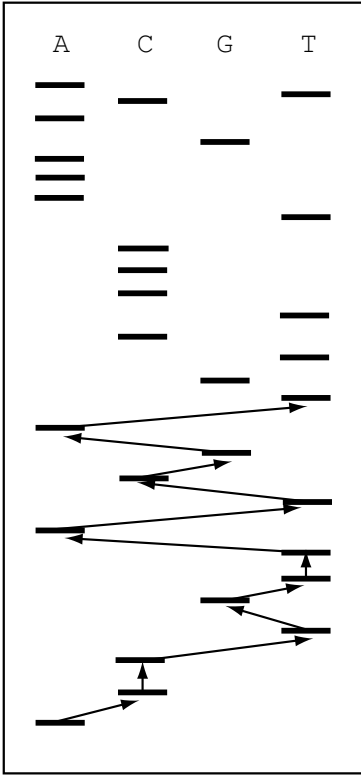
smallest nearer the exit point (bottom). Each band or peak on the gel corresponds to the collection of reaction product molecules that have terminated at the same position on the template, and the spacing between one band or peak and the next is related to the number of nucleotides separating successive A residues in this illustration. (Note the similarity between this concept and that of the Smith-Birnstiel restriction-mapping approach discussed in Chapter 4.) Similar separate sequencing reactions are performed to generate bands terminated at C, G, and T (in the newly synthesized strands), corresponding to the bases shown in lowercase on the template.

For simplicity, we describe the analysis of products by using vertical slab gels. In actual practice, automated capillary sequencers are now employed for large-scale sequencing. If the reaction solutions are loaded onto a gel, one in each of four lanes, after electrophoresis, four sequencing “ladders” corresponding to positions of A, C, G, or T are produced, as shown in Fig. 8.2A. The shortest fragment corresponds to termination near the 3′ end of the primer. Since the DNA sequence is conventionally written 5′ → 3′, we therefore start with the short fragments and read up, switching to whichever lane has a band the next increment up. This is continued until the resolution is poor or until compressions hide the actual number of bands. Currently, this limit is reached after reading 500 to 800 bases. An autoradiogram corresponding to an actual sequencing slab gel (using a different template) is shown in Fig. 8.2B.

With automated sequencing, four different fluorescent dyes are employed, one for each ddXTP employed. In that case, the reaction products can be pooled and resolved in a single lane or channel, as shown in Fig. 8.2C. We

Fig. 8.2 (opposite page). [This figure also appears in the color insert.] Reading DNA sequences. Panel A: Idealized data from four sequencing reactions, with products resolved by electrophoresis on a polyacrylamide slab gel. The sequence being determined corresponds to a portion of the sequencing reaction diagrammed in Fig. 8.1. The top and bottom bands in the “A” lane correspond to the top and bottom products, respectively, listed in Fig. 8.1D. Thin arrows indicate the order in which lower bands are to be noted during the readout of the DNA sequence. Panel B: Results of a sequencing experiment employing a radioactively labeled product detected by autoradiography. Notice how closer band spacing and high band intensities complicate the reading of the upper portions of the gel. Also notice that the lanes are not parallel: those on the right are skewed. Panel C: Graphical display of output from an automated DNA sequencer, with different colors corresponding to differing fluorescent labels on each of the ddXTPs: red = A, green = C, black = G, and blue = T. The abscissa corresponds to the time of electrophoresis. Upper panel: Uncorrected intensity data showing spillover between different color channels. Lower panel: Color-corrected output of intensity data shown in the upper panel. Notice how color correction restores the Cs that were apparently missing in the upper panel at positions 115 and 130. Panel C provided by Prof. Lei Li, University of Southern California.

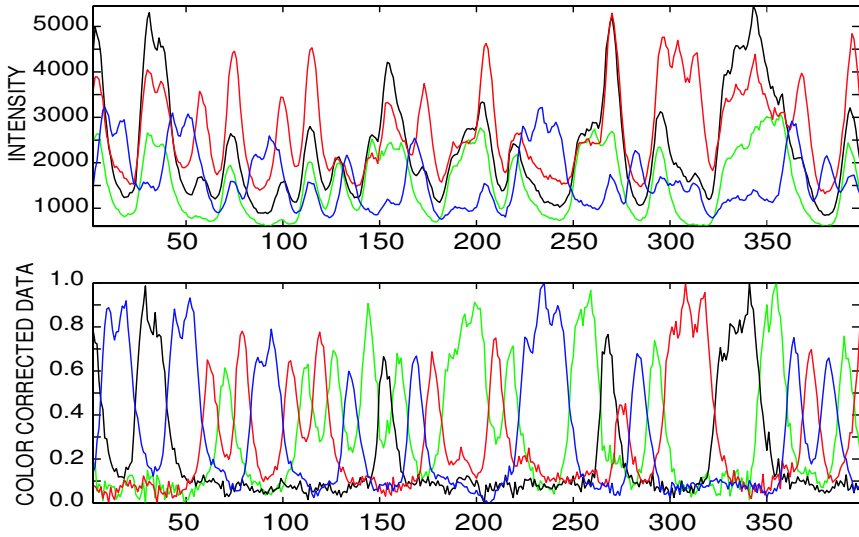
A.



B.



C.



discuss the use of capillary gels in high-throughput sequencing in the next section.

What we need to know for the next section is that sequences may be from either strand and that they will typically be 500-800 bp long. The problem addressed in the next section is how to use a very large collection of sequences from different templates, all derived from the same genomic DNA, to reconstitute the genomic sequence from the collection of individual sequence “reads.”

8.2.3 Analytical Tools: DNA Sequencers

When slab-gel technologies were first employed, the advantages of being able to run many samples in parallel were immediately evident. For sequencing with radiolabeled substrates, however, four lanes were typically employed for each substrate being analyzed (one each for A, C, G, and T). Slab gels typically draw large amounts of current during electrophoresis, which places limits on the voltage that can be applied. Excessive voltages lead to high current, which in turn can heat the gel to the point that the buffer boils or the glass plates surrounding the slab crack. Either of these events ends the experiment. Slab gels are very fragile and hard to handle without tearing; moreover, the procedures required to pour the gels are difficult to automate. Analysis by phosphorimaging or autoradiography is slow and requires an additional digitizing step to locate band positions for eventual analysis. Because measurements are made at a single time point (the end of the “run”), the larger fragments will not have migrated as far as the smaller ones and thus are not resolved as well. In those regions of the gel near the origin, bands are compressed together and unreadable.

Capillary array electrophoresis has alleviated these problems. Instead of a slab gel, a set of gel-filled capillaries (75 μm inside diameter and 40 cm long, for example) is employed. The small inside diameter of the capillaries reduces the current, and the large surface-to-volume ratio improves heat dissipation, so that higher voltages (e.g., 10,000 V rather than 1000 V) can be employed. This allows shorter run times. Unlike in the slab gel approach, the measurements in capillaries are made continuously (i.e., at many time points) at one position—at or near the exit point from the capillary. This means that larger fragments can be resolved over the full length of the capillary, although the longer time required for them to exit the capillary allows broadening of the bands because of diffusion. Capillary sequencers are highly automated with respect to loading samples, recharging the gel matrix in the capillaries, and acquiring data. Depending upon the resolution required, a typical automated sequencer can produce about 600–800 bases of readable sequence in a 1 hour run, and with a turnaround time of less than an hour, it can perform more than 20 such runs per day. With a “typical” 96 capillary array sequencer, this works out to more than $96 \times 700 \times 20 \approx 1,000,000$ bases per day per instrument. Some automated sequencers employ $4 \times 96 = 384$ capillaries, and these instruments have capacities approaching 3,000,000 bases per 24 hour day.

With capillary sequencers, DNA polymerization products employing all four terminators (ddATP, ddCTP, ddGTP, ddTTP) are resolved in the same capillary. Products from each of the four reactions are labeled with a different dye. The readout consists of a fluorescent signal with emission wavelengths characteristic of each type of dye. An example of the output is graphed in Fig. 8.2C. Fluorescent dyes can be attached to the primer (dye primer), in which case four different reactions are required if the same primer sequence is used. The four reaction product solutions are mixed prior to electrophoresis. Fluorescent dyes can also be attached to the ddXTPs (dye terminator), which allows all four sequencing reactions to be conducted in the same reaction tube. Dyes and polymerases have been optimized so that the dye-ddXTP compounds are incorporated into the growing chain with comparable efficiencies. In general, each of the four fluorescent dyes might require a different wavelength for optimal excitation. However, by adding a second “donor” dye and using fluorescence energy transfer to an “acceptor” dye that discriminates between reaction products, it is possible to use a single exciting wavelength appropriate to the donor dye. Excitation is done with a laser that either illuminates samples exiting from all capillaries simultaneously, or scans across the array repeatedly, illuminating the exiting samples individually. The light produced by fluorescence is split into four different wavelength ranges (channels), and the intensity in each channel is recorded as a function of time.

8.3 The Three-Step Method: Overlap, Layout, and Multiple Alignment

Determining the sequence of bases in biological sequences is a long and challenging problem, and an important component is computational. As the previous discussion indicated, the data typically are randomly located reads that are short compared with the target (i.e., the unknown DNA to be sequenced or determined). The orientation (5' to 3' or 3' to 5') relative to a map convention is unknown, and the reads are of good quality but not perfect. In this section, we present the usual method of sequencing, which as noted above is colorfully called shotgun sequencing. There are three computational steps in the process of shotgun sequencing as usually performed: pairwise comparison, **layout**, and **multiple alignment**. Pairwise *overlap alignment* (i.e., alignments involving the ends of two sequence strings) produces scores that are used as indicators of genomic overlap. Those scores can be used to obtain clusters of reads with mutually consistent overlap scores. Finally, the layout is used as a basis for multiple alignment that produces the consensus sequence. It is a mistake to think the entire target sequence will be determined even if the assembly is perfect. The random location of the reads makes the coverage of the target follow the statistical distribution of oceans and islands as de-

scribed in Chapter 4. Each of the three steps is briefly described below, and then a small example is presented for illustration.

If there are n reads, the read set must be augmented by the reverse complements of the DNA sequence from each read so that the set of potential reads is $2n$ in size. The job of pairwise comparison is to look for potential overlaps. This means that for every two reads r and s from the set of n original reads, there are two comparisons: read r versus read s , and read r complemented, r^* , compared with read s . (You should check that this includes all *four* possibilities for the arrangement of read overlaps in the actual genome sequence.) The *Drosophila* whole-genome shotgun assembly had 3×10^6 reads of 500 bases. This means there were approximately 10^{13} comparisons to perform. How difficult are the comparisons? Computational Example 8.1 presents an extension of the local alignment algorithm to handle this problem. In fact, for situations such as the *Drosophila* project mentioned, this algorithm is too costly in time: if each comparison takes time proportional to 500^2 (the cost of a dynamic programming comparison), then the total *time complexity* for all comparisons is proportional to 2.5×10^{18} , which is not practical with current computer technology. Instead scientists have found ways to speed up and shortcut the comparisons.

The overlap alignment algorithm is presented in pseudocode in Computational Example 8.1. It is a modification of the local alignment algorithm, but the “0” in the recursion is left out. The details of the logic are not presented here, but it is similar to the various dynamic programming alignment algorithms (Chapter 6). An alignment matrix (elements O_{ij}) is employed for each alignment. The best overlap according to the scoring scheme is given as the largest number in the rightmost column and bottom row of the alignment matrix.

Computational Example 8.1: Pseudocode for overlap alignment

```

Input sequences A, B
Set  $O_{i,0} = O_{0,j} = 0$  for all  $i, j$ 
for  $i = 1$  to  $n$ 
  for  $j = 1$  to  $m$ 
     $O_{i,j} = \max\{O_{i-1,j} - \delta, O_{i-1,j-1} + s(a_i, b_j), O_{i,j-1} - \delta\}$ 
  end
end
Best overlap =  $\max\{O_{i,m}, O_{n,j}; 1 \leq i \leq n, 1 \leq j \leq m\}$ 

```

Now we turn to the **layout** phase of shotgun sequence assembly. Having a matrix of pairwise comparisons is just the beginning. With the pair of reads r and s , there are four ways they can fit into an assembly: r vs. s , r^* vs. s , r vs. s^* , and r^* vs. s^* . We group all these pairwise comparisons into clusters

based on their scores. If, for example, two reads r and s of length 500 have scores that indicate overlap of approximately 400 bases, then if r has a score indicating a 300 base overlap with a third read t , then t must have at least a 200 base overlap with s ; otherwise the pairwise scores are inconsistent with a genomic layout. Fixing the orientation of one of the pairs of reads determines the orientation for the remaining reads in the cluster.

Finally, there is the problem of determining the **consensus sequence** from the layout. It might be imagined that this is an easy job, and it would be if the sequences were perfectly aligned. Unfortunately, as with many computational problems in biology, this is almost never the case. Instead, sequencing errors, fragments incorrectly clustered, and the approximate nature of the initial alignment all make this problem most challenging. A **greedy procedure** (which recursively “grabs” solutions close to a local optimum; see www.nist.gov/dads/HTML/greedyHeuristic.html) allows an initial alignment, if desired, in the following manner. The highest-scoring overlap can be fixed and the sequences aligned. Then the sequence with the highest overlap score with any member of this cluster can be added, and so on. The identity of the base at any position is then taken to be the majority letter in the resulting multiple alignment. Of course, this is only the beginning of producing a consensus sequence since the alignment is very unlikely to be correct.

To illustrate these ideas, we take a “toy” example through the assembly process. The DNA to be sequenced is short (70 bases), and the reads are 8 bases long. This sequence is taken from an earlier example (Section 2.1) along with enough 3′ and 5′ sequences (7 bases) on both ends, underlined in the example below, so that the full 70 bases can be covered (if reads overlap the ends). The reads are random in position and orientation, and the problem is so small that we must take 100% accuracy in our reads. As we will see, assembly is still not an entirely easy task. The DNA represented by the “top strand” sequence is

5′ – CAGCGCGCTGCGTGACGAGTCTGACAAAGACGGTATGCGCATCGTGATTGAAGTG
AAACGCGATGCGGTTCGGTGAAGTTGTGCT – 3′

Table 8.1 gives the 20 reads and their reverse complements (5′ to 3′ orientation for both). The reads, which are a subset of all possible reads, were chosen at random locations in the sequence and in random orientations (i.e., some correspond to the *complement* of the strand shown).

Next, we present a 20×20 overlap matrix (Fig. 8.3) that indicates which fragments or their complements overlap by more than a prespecified number of letters. In location (k, l) with $k < l$, the entry is the number of bases of overlap for r_k versus r_l , and if $k > l$ the entry is the number of bases of overlap of r_k with r_l^* . No diagonal entries are made because alignments of a sequence with itself are uninformative. If the best overlap is less than 3, it is not entered. This is because, for iid letters, overlaps of two random letters happen 1/2 the time, and two-letter overlaps happen 1/8 of the time. (This calculation

Table 8.1. Twenty reads and their reverse complements

No.	Read	Read*
1	CATCGTGA	TCACGATG
2	CGGTGAAG	CTTCACCG
3	TATGCGCA	TGCGCATA
4	GACGAGTC	GACTCGTC
5	CTGACAAA	TTTGTCAG
6	ATGCGCAT	ATGCGCAT
7	ATGCGGTC	GACCGCAT
8	CTGCGTGA	TCACGCAG
9	GCGTGACG	CGTCACGC
10	GTCGGTGA	TCACCGAC
11	GGTCGGTG	CACCGACC
12	ATCGTGAT	ATCAGCAT
13	GCGCTGCG	CGCAGCGC
14	GCATCGTG	CACGATGC
15	AGCGCGCT	AGCGCGCT
16	GAAGTTGT	ACAAC TTC
17	AGTAAAAC	GTTTCACT
18	ACGCGATG	CATCGCGT
19	GCGCATCG	CGATGCGC
20	AAGTAAAA	TTTCACTT

is for the two ends, doubling the chances. Actually, the probability of a two-letter overlap is $1/8 - 1/256$; see Exercise 2.) Considering that there is also another reversed comparison, effectively doubling this probability, two-letter overlaps occur too frequently to be of interest. Three-letter overlaps happen between two fragments by chance with probability approximately $2 \times 1/64$. This probability and an equal number for the reversed comparison imply that there will be approximately one spurious overlap per fragment. This noise level we can handle. Making the overlap requirement four letters would restrict us from assembling much sequence, although what we would produce would likely be accurate.

Now that we have the overlap matrix (from the pairwise comparison step), we turn to the layout and alignment steps. As you will see, in this small example we combine them into one process. First, we examine the matrix for significant overlaps. Obviously, read 1 has two substantial overlaps of 7 bases with reads 14 and 12. They are

GCATCGTG	14
CATCGTGA	1
ATCGTGAT	12

This layout and alignment is completely consistent and believable. We are beginning to assemble the DNA text. Next, we incrementally add overlaps to

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
*						3						7		7						5
	*									6	5									3
		*				7							3						5	
			*					4												
				*																
3		7				*							4					3	6	
3							*			3	4									
								*					5							
									*				3					3		
			3							*	7									
											*									
												*	6							4
		4											*	5						3
						4	4							*						6
													5		*					
																*				
																	*			7
						3			4									*		
						6							4							*
																				*

Fig. 8.3. Overlap matrix for sequence assembly. Rows and columns correspond to sequence reads, where the numerical entries are the number of positions (or score) of the match (overlap) between the reads. Only scores greater than or equal to a threshold of three are shown.

the array in a greedy fashion, including reads 19, 6 (or 6*), 7*, and 13*. The alignment is

<u>G</u> ACCGCAT	7*
<u>A</u> TGCGCAT	6=6*
GCATCGTG	14
CATCGTGA	1
ATCGTGAT	12
<u>G</u> CGCATCG	19
<u>C</u> G <u>C</u> A <u>G</u> C <u>G</u> C	13*
CGCATCGTGAT	

and we have determined 11 bases of sequence. Underlined bases at the left ends of reads 6, 7*, 19 and 13* are ambiguous, and it is now easy to see

why the process of sequencing is challenging. We have good evidence for the sequence 11 bp and cannot call the other letters with any confidence, even with error-free sequencing.

Moving on to the overlap from reads 10 and 11, by the same greedy technique, we obtain

CGGTGAAG	2
GTCGGTGA	10
GGTCGGTG	11
ATGCGGTC	7
<hr style="width: 100%;"/>	
ATGCGGTCGGTGAAG	

Since we have no contradictions, we use the entire span of 15 bases. Finally, we identify overlaps for reads 17 and 20.

AGTAAAAC	17
AAGTAAAA	20
<hr style="width: 100%;"/>	
AAGTAAAAC	

As there are no other consistent overlaps (the 10-4 overlap is not compatible with the others), we only get a 9 base determination.

This small example shows us how a sequence can be incrementally assembled from smaller fragments and illustrates some of the difficulties of assembly. In addition, only part of the sequence is determined from these data. After all, the coverage is $20 \times 8/70 = 2.28$. At a coverage of 10 we would with reasonable probability recover most of the 70 bp.

8.4 High-Throughput Genome Sequencing

Shotgun sequencing approaches, including the whole-genome shotgun approach, are currently a central part of all genome-sequencing efforts. These methods require a high level of automation in sample preparation and analysis and are heavily reliant on the power of modern computers. In this, section we present a number of facets of these approaches.

There is an interplay between substrates to be sequenced (genomes and their representation in clone libraries), the analytical tools for generating a DNA sequence, the sequencing strategies, and the computational methods. The key underlying determinant is that we can obtain high-quality continuous sequence reads of up to 500 to 800 bases with current technology. This represents a tiny fraction of either a prokaryotic or eukaryotic genome. The sequence reads are the primary data. As indicated above, the computational problem in large measure is defined by the need to assemble a larger whole from a large number of small parts. DNA reads come from clones containing different insert sizes. Because eukaryotic genomes contain repeated sequences that may be longer than the average sequence read, strategies employing different sizes of cloned inserts have been used to produce an unambiguous sequence assembly. The strategies used will, in turn, determine what fraction

of each clone is sequenced and the coverage of the genome required for each type of clone.

8.4.1 Computational Tools

The problems of sequence assembly were illustrated in Section 8.3. It is beyond the scope of this book to discuss at length the details of sequence assembly software. We can, however, indicate some of the typical features. We start with processing the data that are produced by the sequencing instruments.

Given automated sequencers that produce sequences at the rate of 0.5 to 1.0 megabases per day, clearly **base calling** must be automated. Base calling is the process of identifying which base corresponds to each position in a sequence read. The sequence traces produced experimentally are not perfect: they depend upon template quality and purity, reaction parameters, and the particular sequence of the template. For example, inverted repetitions, particularly in regions having high levels of G residues, can cause peaks to migrate anomalously, leading to *compressions* (individual peaks migrating together as an unresolved, larger peak). Also, sometimes the template may have homopolymer “runs” (TTT \cdots T, for example) at which “slippage” of the polymerase can occur. This may lead to a manifold of additional low-intensity bands. Short fragments may produce bands having anomalous mobility because of the effects of dyes attached to terminating ddXTPs. There sometimes may be “spillover” of light from the emission spectrum of one dye into the wavelength range of another, causing the instrument to report peaks in more than one channel at a particular position. Some of these types of problems are illustrated in Fig. 8.2C.

Commercial sequencing instruments are typically “bundled” with base-calling software. The independently developed base-calling application Phred illustrates the required features of such an application (Ewing et al., 1998). The trace processing portion of Phred proceeds through four different steps. First, it determines idealized predicted peak locations for a given trace based upon peaks that appear to have regular spacing. Second, it identifies observed peaks as those in the trace that exceed a minimum threshold peak area. Some of these are multicomponent peaks that will eventually be split and assigned in the third step. Third, observed peaks are matched to predicted locations. Those peaks that have aberrantly large areas compared with their neighbors are split into two or more peaks that are assigned to the predicted peak locations. Finally, missing peaks are accounted for from among previously uncalled peaks. A very important feature of Phred is that it associates with each base a probability p that the base call is in error. The probability p depends upon things such as peak spacings, peak resolution, and areas of uncalled peaks. The quality of each base call is described by the quality score Q , which is defined as $Q = -10 \log_{10} p$. For example, if the probability that a particular base is called in error is 0.001, the quality score Q is 30.

Once the output of the sequencing machines has been written into data files, the data are ready for assembly. There are a number of sequence assemblers available. A common assembler often used together with Phred is Phrap (“*ph*ragment assembly program” or “*phil*’s revised assembly program:”; Green, 1999). Others are the CAP assemblers (a recent version, CAP3, is available from Huang (Huang and Madan, 1999)), the TIGR Assembler (The Institute for Genome Research (TIGR), 1995), and the Whole-genome Assembler used by Celera (Myers et al., 2000). For references to other assemblers such as EULER and ARACHNE, see Venter et al. (2003). Sequence assemblers have differing levels of complexity, but there needs to be provision somewhere in the assembly pipeline for the following processes:

- Screening out vector sequences or chimeric reads;
- Trimming off unreliable base calls from each read (at both 3’ and 5’ ends);
- Computing overlaps between pairs of reads, using the highest-quality portions of each read;
- Screening out doubtful overlaps (i.e., those not having minimum length, minimum percentage identity, minimum similarity score, or having too many discrepancies in areas where the base quality is high);
- Constructing contigs; and
- Producing a consensus sequence by multiple sequence alignment with reliability scores at each position.

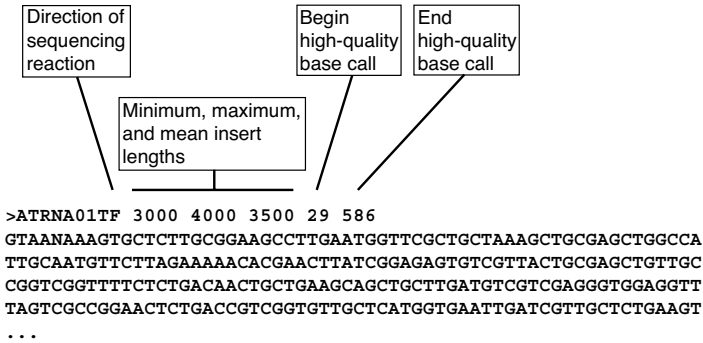
The input data to the assembler include the sequence files for each read and the corresponding files of quality scores. Examples of such files for the TIGR assembler are shown in Fig. 8.4. By providing the quality scores in a second file, the assembler can use high-quality base calls even if there are a few low-quality ones farther in from the ends. The alternative is aggressive trimming of the sequence to include only regions composed exclusively of high-quality base calls. This would shorten the read length and correspondingly increase the required clone coverage. Parameters that must be supplied to the assembler are things such as (TIGR, 1995):

- Minimum length of overlap between pairs that will be considered for assembly;
- Minimum percentage of identity within overlap of two fragments to be considered for assembly;
- Length of oligomers used in rapid similarity search;
- Information parameters for each read (e.g., clone identification and whether it is a forward or reverse read).

The output is the consensus assembled sequence with the reliability score at each position.

The actual computational resources for performing sequence assembly can be substantial, for reasons indicated in Section 8.3. The banded-search approach similar to that used in FASTA speeds the computation of overlaps, but the assembly still can be time-consuming. In 2001, the human genome

A. Sequence file



B. Quality file

```

>ATRNA01TF
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 15 00 00 00 00 00 00 00 25 23 33 31 23 23
00 00 00 28 28 23 23 23 23 23 23 45 40 37 32 28 28
19 32 45 38 37 18 18 00 00 00 00 25 34 36 36 34 34
34 34 31 31 31 34 34 34 37 37 41 41 45 45 37 37 37
27 37 37 40 40 34 34 34 34 37 37 37 40 45 37 34 34
.
.
.
25 27 25 30 25 23 22 18 21 23 26 26 33 35 22 18 00
00 00 00 15 26 23 18 00 00 00 00 18 21 30 30 30
34 37 37 34 34 33 33 28 28 28 37 32 32 19 19 19 30
22 19 00 00 00 21 25 37 37 37 37 37 37 37 37 37 37
.
.
.
34 38 27 21 00 00 00 21 20 26 29 29 31 29 26 24 15
00 00 15 00 00 23 22 18 00 00 00 17 00 00 00 00 00
00 00 00 00 00 00 00 18 19 27 23 00 00 21 21 00 00
.
.
.
00 00 20 21 15 15 17 17 26 23 25 18 28 17 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 17 00 00 00
00 00 00 00 00 00 00 00 00 00 00 16 17 26 24 21 24
19 17 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 16 16 15 21 17 15 00 00 00 17 00 00 00 00 00 00
00 00 00 00 00 00 00 15 15 00 17 00 00 00 00 00 00
00 00 00 00 00 00 00 00 19 00 00 00 00 00 00 00

```

Fig. 8.4. Sequence and quality files used in sequence assembly. The sequence file (panel A) is in FASTA format. Relevant parameters are listed in the header line of the sequence file. The quality file (panel B) contains a *Q* score for each position listed in the sequence file. “00” indicates base calls that are unreliable. The figure shows only portions of the quality file, taken at increasing distances from the position of the primer. Reproduced and reprinted, with permission, from The Institute for Genomic Research (TIGR). Copyright ©2003 The Institute for Genomic Research (TIGR). For details, visit <http://www.tigr.org/software/assembler/helpfile.html>.

sequence assembly required 20,000 hours of CPU time and 500 GB of storage, with the use of forty, four-processor machines, each having 4 GB of RAM, running in parallel. Half of this time was employed in computing the overlaps between reads (Venter et al., 2002, 2003).

8.4.2 Genome-Sequencing Strategies

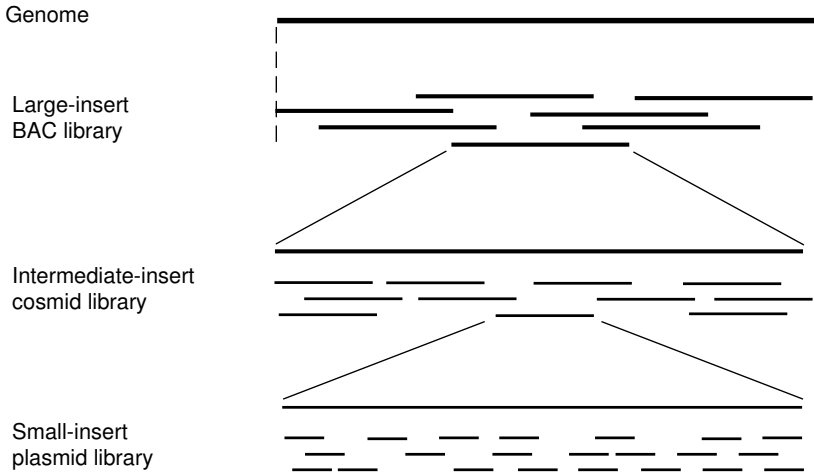
As we indicated earlier, the type of data and available computational resources dictate which strategies are feasible. All strategies require cloning, and strategies commonly employ shotgun sequencing at some level. The differences in strategies lie in the use of clone mapping, and the point at which random shotgun sequencing is initiated. Three particular strategies are: the clone-by-clone shotgun approach, shotgun sequencing of BACs joined into a minimum tiling path by sequence-tagged connectors, and whole-genome shotgun (WGS) assembly. These three methods are illustrated in Fig. 8.5. For bacterial genomes, the WGS method alone is often sufficient. For larger genomes, elements of all three approaches can be blended in hybrid strategies. For a good review of these strategies, see Green (2001).

The whole-genome shotgun assembly (Fig. 8.5C) was first employed with viral genomes (Anderson, 1981; Gardner et al., 1981; Sanger et al., 1982). As we indicated earlier, a large number of randomly selected small-insert clones are used to generate sequences at appropriately large levels of **sequence coverage**. Sequence coverage is the average number of times any given genomic base is represented in sequence reads. This worked quite well for small, uncomplicated genomes such as the cauliflower mosaic virus (8031 bp) and bacteriophage lambda (48,502 bp). This method also has been employed for cloned fragments from larger genomes in the clone-by-clone shotgun approach. At the inception of the Human Genome Project, a WGS approach to the entire genome (i.e., sequence many randomly-chosen clones without any prior physical or genetic mapping) did not appear to be feasible. This was because of the known abundance of repeats that would lead to ambiguous assemblies and the limitations in sequence production and computing.

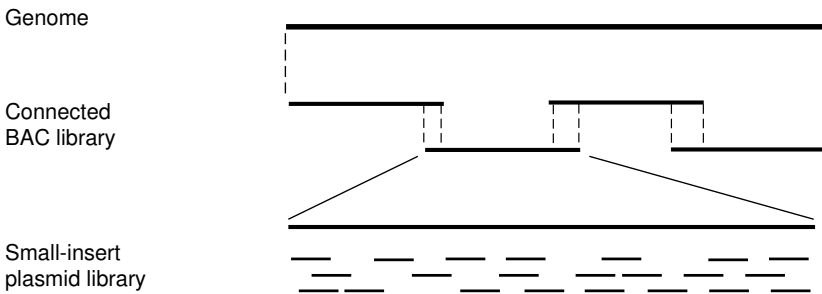
When the Human Genome Project was first proposed, it was assumed that some aspects of a top-down approach would be required in a “divide and conquer” strategy represented by the clone-by-clone shotgun approach (Fig. 8.5A). The idea was to construct a high-resolution *genetic* map, a low-resolution physical map from large-insert clones (originally YACs but later BACs), and a high resolution physical map based on cosmids. Cosmids in a minimum tiling path could then be used as substrates for random shotgun sequencing.

An intermediate approach that avoids the need for mapping (Fig. 8.5B) employs large-insert BAC clones and sequence-tagged connectors (Venter et al., 1996). For this method, the entire genome is cloned into a BAC library, and the ends of the BACs are sequenced. The BAC end sequences are the sequence-tagged connectors, so-called because these sequences can be used to

A. Three-stage divide-and-conquer



B. Shotgun sequencing of connected BACs



C. Whole-genome shotgun sequencing

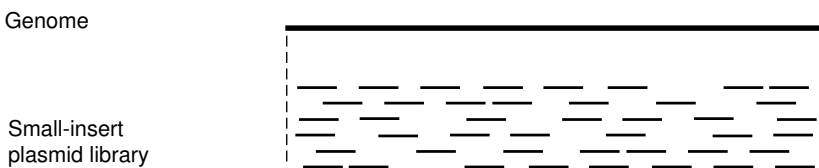


Fig. 8.5. Strategies for genomic sequencing. Panel A: Clone-by-clone three-stage divide-and-conquer approach. Panel B: Sequencing of BACs joined by sequence-tagged connectors (STCs). Panel C: Whole-genome shotgun sequencing.

identify other BACs that overlap sequenced ends. For example, PCR reactions designed to amplify unique sequences at the ends of a BAC can be used to screen the BAC library for other clones that overlap the ends. Combined with restriction digest fingerprinting, this process can generate a minimum tiling set of BAC clones, as discussed in Section 4.6. Each BAC clone (insert size about 150 kb) can then be subjected to random shotgun sequencing.

In actual practice, hybrid strategies are useful, especially for model organisms for which prior genetic mapping and cloning have been undertaken. Hybrid strategies can blend (in different proportions) the clone-by-clone shotgun and the WGS approaches. For example, we could use the WGS method to provide small contigs at high levels of sequence coverage, and these could be further assembled based upon partial assemblies of large-insert clones that perhaps had been shotgun-sequenced at lower coverage (see below). Because a clone-by-clone shotgun approach will usually have generated a minimal tiling set of clones, hybrid strategies employing this approach will reduce the likelihood of unfilled gaps in the sequence that might result from using the WGS approach exclusively. The optimal proportion of clone-by-clone to WGS approaches will depend upon the genome.

The assembly of the brown Norway rat genome (Rat Genome Sequencing Project Consortium, 2004) is a good example of one hybrid strategy. The several operations were actually conducted in parallel, but conceptually they included the following steps. A set of BAC contigs was generated based upon fingerprinting of the almost 200,000 BACs. The resulting fingerprint contig map was used to select a subset of BACs that were *individually* shotgun sequenced at *low* sequence coverage (depth approximately 1.8). These sequence reads are thus binned in the genome region contained in the corresponding BAC insert, but the read depth is not sufficient for assembly of the whole BAC insert sequence. In parallel, however, WGS sequencing to a depth of $7\times$ sequence coverage had been performed. The low-coverage sequences binned in each BAC could be used as probes to “fish” for overlapping sequences in the larger set of WGS sequencing reads. The resulting set of reads belonging to each BAC (binned low-coverage reads + overlapping reads identified among the higher-coverage WGS sequence reads) were then assembled together to produce a set of “enriched” BACs, or “eBACs.” Based upon sequence overlaps, the set of eBACs were then assembled into contigs (“bactigs”) and into higher-order assemblies (scaffolds; see below) in the usual way.

8.4.3 Whole-Genome Shotgun Sequencing of Eukaryotic Genomes

Whole-genome shotgun sequencing clearly is a viable approach for small genomes, such as viral genomes. But it was not clear that this approach would be feasible for larger genomes. In particular, there were doubts that it was appropriate for vertebrate genomes in general and the human genome in particular (Green, 1997). Part of the reason for this pessimism had to do with the

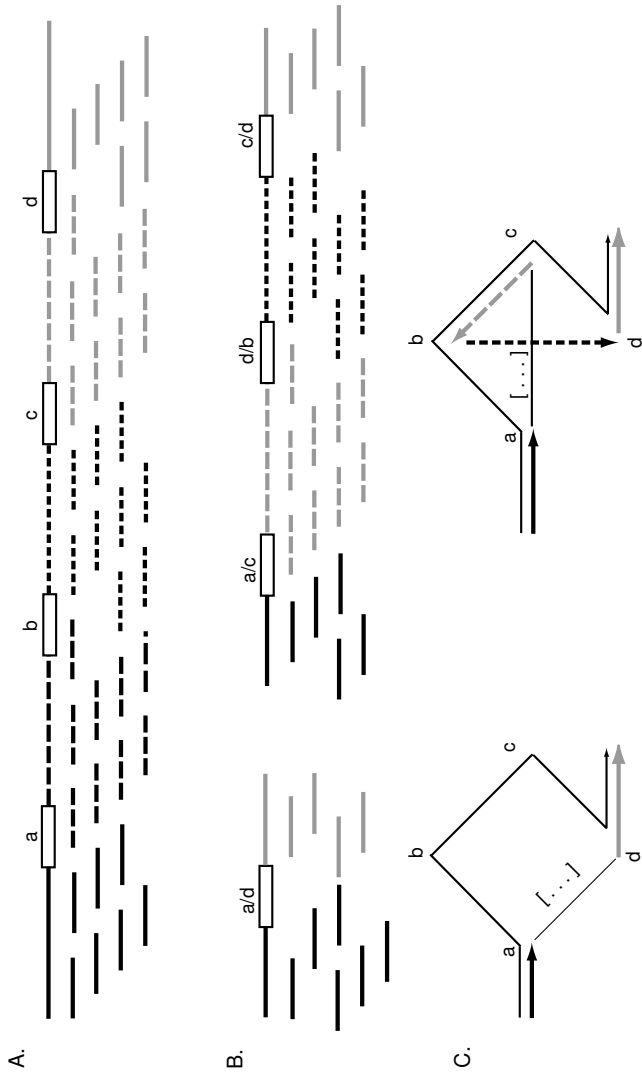


Fig. 8.6. Alternative assemblies resulting from the occurrence of repeated sequences. Panel A: The repeated sequences *a*-*d* are separated by unique sequences (solid or broken lines having different shading). The repeated sequences are too long to be spanned by the sequence reads (short lines underneath). Panel B: Assemblies consistent with the sequence reads may delete the entire block of repeated sequences (except for a single composite repeat *a/d*) as shown on the left, or may delete or permute portions of the sequence between *a* and *d* as shown on the right. Panel C graphically illustrates the assemblies in panel B. The correct assembly (*a*-*b*-*c*-*d*) corresponds to one particular solution to the Hamiltonian path problem. Both assemblies in panel B delete portions of the sequence (thin line labeled with “[...]”) and therefore fail to visit all four vertices. In the assembly at the right, vertices visited in the incorrect order result in permutation of two stretches of unique sequence that are included in the assembly.

nature of eukaryotic genomes. It had been known for a long time that eukaryotic genomes contain repeated sequences. Physical biochemical methods had established that many copies of several families of repeated sequences were present in *Drosophila* and human genomes. For example, the human genome consists of about 45% repeated sequences, with over a million copies of *Alu* elements alone (International Human Genome Sequencing Consortium (IHGSC), 2001). This creates a problem for DNA sequence assembly, as illustrated in Fig. 8.6.

We imagine a region of the genome containing four copies of a repeated sequence, labeled a, b, c, and d (Fig. 8.6A). The repeats may not be precisely identical in their sequence but have diverged over time to various extents from a consensus sequence. When random shotgun sequencing of this genome segment is performed, the resulting fragments no longer indicate the connectivity of the unique sequences between the repeated sequences. This is because some repeats are sufficiently long that they cannot be completely spanned by a sequence read. In attempting to reconstitute the original sequences from the individual fragments, we find that there are a number of possible ways of assembling the sequence. Some of these connections may “collapse” a portion of the genome, deleting unique sequences that should have been included in the assembly (Fig. 8.6B). The problem is redrawn as a **Hamiltonian path problem** in Fig. 8.6C. The Hamiltonian path problem is to find a path through a graph such that each vertex (a repeated sequence in this case) is visited only once. There are no efficient algorithms for solving the Hamiltonian path problem, which is NP-complete (See Section 4.3). We illustrate in Fig. 8.6C the correct path (thin line) and incorrect paths (thick lines or paths that delete some unique sequences) for the examples in Fig. 8.6B. How then can we hope to perform a whole-genome sequence assembly from a random shotgun approach if there can be 10^5 to 10^6 vertices?

We saw already in the case of the double-digest problem (Chapter 4) that difficult computational problems sometimes can be avoided by imaginative experimental designs (e.g., the Smith and Birnstiel mapping method). Altered experimental designs also made it possible to apply WGS approaches to eukaryotic genomes. The first eukaryotic test case for this method was the assembly of the *Drosophila melanogaster* genome (Meyers et al., 2000), and the methods developed there have been extended to the human genome (Venter et al., 2001). The key experimental feature of these approaches is the use of multiple clone libraries, each having different sizes of inserts, and with tracking of sequences from *both* ends of each cloned insert.

We need to step back for a moment to recall the approaches used in the first WGS sequencing of a free-living organism, *Haemophilus influenzae* (Fleischmann et al., 1995). Bacterial genomes are easier subjects for WGS sequence assembly because they are about 1/100 to 1/1000 the size of typical vertebrate genomes and because they have few repeated sequences. Even for *H. influenzae*, two different libraries were employed: a small plasmid library with size-selected inserts approximately 2 kb in size and a lambda library with inserts

about 15–20 kb in size. The sequence assembly was aided by the following experimental design features:

- Two libraries—one small-insert and one larger-insert—were employed.
- Sequences from both ends of the inserts were obtained, and these end-sequence pairs were tracked for use in assembly.
- The insert sizes in the libraries were confined to a relatively narrow size range.

The small-insert library provided the templates for generating the high levels of sequence coverage necessary to include most of the genomic sequences. The large-insert library aided in closing gaps in the sequence, and, most importantly, the end reads from the large inserts, when each lay in a different contig, provided data on the distance and orientation of these contigs relative to each other.

These approaches were extended and refined in their application to the *Drosophila* genome (Adams et al., 2000; Meyers et al., 2000) and the human genome (Venter et al., 2001). The methods had to be refined to accommodate the larger fraction of repeated sequences (about 3.1% in the *Drosophila* genome and about 45% in the human genome). We use the details for the *Drosophila* project because they are simpler but typical of both. Three different libraries were prepared for the *Drosophila* sequencing project: a 2 kb insert high-copy-number plasmid library (8× clone coverage); a 10 kb insert low-copy-number plasmid library (7× clone coverage); and a 130 kb insert BAC library (13× clone coverage). (This may be a good time to review the concept of coverage, discussed in Section 4.5.1.) In most cases, sequence reads were obtained from both ends of each insert. Such reads are called **mate pairs** or **paired-end sequences**. In total, the raw data consisted of 3.2 million reads plus the clone and mate information for each. Note that there are two types of coverage. **Clone coverage** corresponds to the average number of genome equivalents contained within *the collection of complete inserts* represented in each library. Clone coverage need not be the same for the different libraries. As we indicated earlier, sequence coverage corresponds to the average number of times *each base position* is represented in the collection of all sequence reads. Note that sequence coverage is less than clone coverage because only a fraction of each insert is represented in the mate pairs. Before sequences of mate pairs are used for assembly, they are trimmed to leave only sequences with sufficient quality, and they are screened to remove contaminating sequences (e.g., vector sequences or *E. coli* sequences) and possibly to set aside reads from some of the repeated sequences.

The first phase of the assembly is alignment of sequence reads. The alignments must meet predetermined criteria. For example, *Drosophila* and human sequence assemblies required overlaps that were at least 40 bases in length and 94% identical. The products of this initial “overlap” phase of sequence assembly are **unitigs**. A unitig is a small contig composed of sequence reads that have been unambiguously assembled. (The overlaps are uncontested—

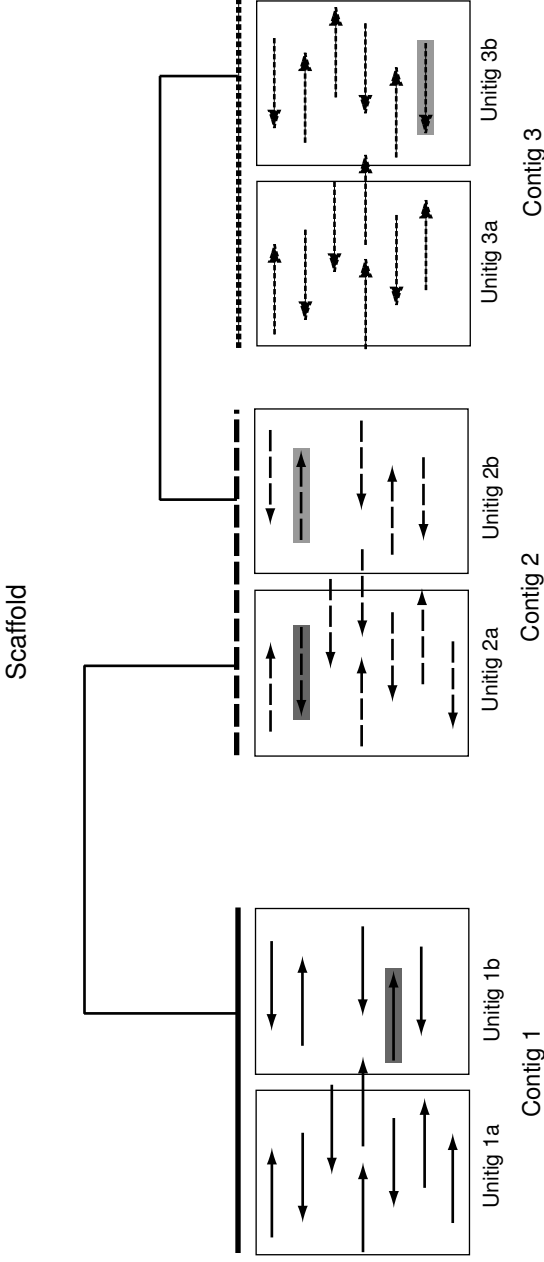


Fig. 8.7. Anatomy of a scaffold. Sequence reads are indicated by lines with arrowheads. Based upon pairwise comparisons, sequence reads can be assembled into small contigs called *unitigs*, which can further be merged into larger *contigs* based upon sequence reads present as mate pairs on small- and intermediate-insert clones. Contigs are completely spanned by a DNA sequence, although not all regions may have the same depth of coverage. Contigs can be further grouped to form a *scaffold*. Even though the regions between these contigs may not be represented by a DNA sequence, large-insert clones whose mate-pair reads lie in different contigs (shaded boxes) allow those contigs to be correctly positioned and oriented.

there are no contradictory overlaps.) In the human sequence assembly, each unitig consisted of about 30 overlapping sequence reads. Some of these unitigs are a result of assembling multiple copies of repeated sequences. These “over-collapsed” unitigs are recognized by significantly larger-than-average **read depth** or sequence coverage, and they are not used in the assembly. The remaining unitigs are called **U-unitigs**, which contain *unique* sequences, although they may include repeated sequences at their ends.

The next phase is to assemble U-unitigs into larger contigs. Remember that contigs are continuous stretches of sequence; therefore, unitigs and U-unitigs are contigs. **Contigs reported by the sequence assembler** are larger contigs composed of U-unitigs that have been placed and oriented based upon mate pairs from clones that “bridge” between them (see Fig. 8.7). These larger contigs, in turn, are assembled into scaffolds. A **scaffold** is a set of contigs that have been ordered and oriented such that the approximate spacing between them is known. Again, the mate pair reads, particularly those from the larger-insert clones, are used to produce the scaffolds. There still may be gaps resulting from a region of the genome not sequenced or from repeated sequences that were set aside earlier during the identification of U-unitigs.

The ultimate goal of a genome sequence assembly is a **finished sequence**—an ungapped listing of all consensus bases in the genome together with a high-quality score for each base. In practice, we may need to be content with an assembly of euchromatin only since clones containing heterochromatin may not be stable. The initial published genome sequence of *Drosophila* represented only 120 Mb of the 180 Mb genome because 1/3 of the *Drosophila* genome is heterochromatin. The initial version of a genome sequence, performed at low sequence coverage, is called a **draft sequence**. A **prefinished sequence** is a consensus assembly with gaps and some low-quality consensus scores, but with the desired high level of sequence coverage. Production of a finished sequence from a prefinished sequence is a slow process specifically tailored (not automated) to the particular deficiencies of each problematic region in the prefinished sequence. Filling the gaps may require *directed* approaches such as *chromosome walking*. Regions having low-quality scores may require resequencing either by using appropriately placed custom-synthesized primers or perhaps by using altered sequencing substrates (e.g., 7-deaza dGTP or dITP) to alleviate peak compression problems.

We mentioned earlier the difficulties posed for genome sequence assembly by repeated sequences. We indicated that the repeated sequences corresponded to vertices in a Hamiltonian path problem, that is NP-complete. From the subsequent experimental description, it should be evident how problems with repeats were avoided by experimental design. By focusing on U-unitigs, we focused on *edges*—not *vertices*. Most of these edges are unique sequences. By using large-insert clones, it is possible to correctly orient and place the edges into scaffolds. The edges (U-unitigs) may include portions of flanking repeated sequences at their ends, so they provide guidance for the process of adding back the repeated sequences that may have been set aside either be-

cause of initial screening or as collapsed unitigs. The experimental approaches transformed the problem from one that was difficult to solve computationally to one that was amenable to brute force, large-scale approaches.

Whole-genome shotgun sequencing is now widely used for determining the DNA sequences of both prokaryotic and eukaryotic genomes. As described in this chapter, libraries are used to provide sequences from both ends of the genomic inserts. Using different length inserts allows many of the problems of repeated sequences in the genome to be solved. It is no surprise that DNA containing long nearly-identical repeats may not be properly assembled. In She et al. (2004) it was shown that repeats longer than 15 kb and over 97% similar were not always resolved; therefore, the whole genome shotgun sequencing produced an assembly shorter than the true genome. They recommend a mixed strategy of first a whole-genome shotgun sequence assembly to produce a draft sequence, followed by BAC analysis. An example of such a hybrid strategy was discussed in Section 8.4.2.

References

- Adams MD et al. (2000) The genome sequence of *Drosophila melanogaster*. *Science* 287:2185–2195.
- Anderson S (1981) Shotgun DNA sequencing using cloned DNaseI-generated fragments. *Nucleic Acids Research* 9:3015–3027.
- Ewing G, Hillier LD, Wendl MC, Green P (1998) Base-calling of automated sequencer traces using Phred. I. Accuracy assessment. *Genome Research* 8:175–185.
- Fleischmann RD, Adams MD, White O, et al. (1995) Whole-genome sequencing and assembly of *Haemophilus influenzae* Rd. *Science* 269:496–512.
- Gardner RC, Howarth AJ, Hahn P, Brown-Luedi M, Shepherd RJ, Messing J (1981) The complete nucleotide sequence of an infectious clone of cauliflower mosaic virus by M13mp7 shotgun sequencing. *Nucleic Acids Research* 9:2871–2888.
- Green ED (2001) Strategies for the systematic sequencing of complex genomes. *Nature Reviews Genetics* 2:573–583.
- Green P (1997) Against whole-genome shotgun. *Genome Research* 7:410–417.
- Green P (1999) Documentation for Phrap and Cross_Match (Version 0.990319). <http://www.phrap.org/phrap.docs/phrap.html>.
- Huang X, Madan A (1999) CAP3: A DNA sequence assembly program. *Genome Research* 9:868–877.
- International Human Genome Sequencing Consortium (2001) Initial sequencing and analysis of the human genome. *Nature* 409:860–921.
- Maxam AM, Gilbert W (1977) A new method for sequencing DNA. *Proceedings of the National Academy of Sciences USA* 74:560–564.
- Myers EW, Sutton GG, Delcher AL, et al. (2000) A whole-genome assembly of *Drosophila*. *Science* 287:2196–2204.

- Rat Genome Sequencing Project Consortium (2004) Genome sequence of the brown Norway rat yields insights into mammalian evolution. *Nature* 428:493–521.
- Sanger F, Coulson AR, Hong GF, Hill DF, Peterson GB (1982) Nucleotide sequence of bacteriophage lambda DNA. *Journal of Molecular Biology* 162:729–773.
- Sanger F, Nicklen S, Coulson AR (1977) DNA sequencing with chain terminating inhibitors. *Proceedings of the National Academy of Sciences USA* 74:5463–5467.
- She X, Jiang Z, Clark RA, Liu G, Cheng Z, Tuzun E, Church DM, Sutton G, Halpern AL, Eichler EE (2004) Shotgun sequence assembly and recent segmental duplications within the human genome. *Nature* 431:927–930.
- The Institute for Genome Research (2000) Description, notes, annotated TIGR.Assembler help file, and file format overview. See <http://www.tigr.org/software/assembler/helpfile.html>
- Venter JC, Adams MD, Meyers EW, et al. (2001) The sequence of the human genome. *Science* 291:1304–1351.
- Venter JC, Levy S, Stockwell T, Remington K, Halpern A (2003) Massive parallelism, randomness, and genomic advances. *Nature Genetics* 33:219–227.
- Venter JC, Smith HO, Hood L (1996) A new strategy for genome sequencing. *Nature* 381:364–366.

Exercises

Exercise 1. The first four lanes (reading left to right) in Fig. 8.2B correspond with sequencing reactions (for a single template) supplied with ddA, ddC, ddG, and ddT, respectively. What is the DNA sequence indicated by these four lanes?

Exercise 2. Assume that we are given sequences composed of independent letters of uniform probability. Two sequences overlap by three letters in two different configurations. Show that the probability of at least one three-letter exact overlap is

$$2 \left(\frac{1}{4} \right)^3 - \left(\frac{1}{4} \right)^6.$$

Exercise 3. For $A = \text{CAAACGTCT}$ and $B = \text{AGGCTAAA}$, perform overlap alignment as in Computational Example 8.1 (by hand) using +2 for match, –1 for mismatch, and –2 for every indel letter. Show the matrix and all optimal overlap alignments.

Exercise 4. Show by example that comparison of read r and r complemented (r^*) versus read s includes all of the four possible comparisons of r and r^* with s and s^* .

Exercise 5. In Fig. 8.6C, two alternative assemblies of a genomic segment containing four repeated sequences are diagrammed graphically.

- For this example, calculate the total number of possible assemblies consistent with the sequence reads.
- For the case in which reads from *all* of the unique regions are included, the assembly can be described as a Hamiltonian path problem. Diagram all possible assemblies that begin at *a* and end at *d*.

Exercise 6. A whole genome of 150 Mb is to be sequenced. A small-plasmid library (average insert size 3 kb) providing $5\times$ insert coverage (= clone coverage) and a BAC library (average insert size 150 kb) providing $15\times$ insert coverage are prepared for sequencing. If sequencing reads of length 700 nt are generated from both ends of all inserts in both libraries, compute the average sequence read coverage for the genome.

Exercise 7. The coverage of a sequencing project is the mean λ of a Poisson distribution. It turns out that the Poisson random variable is the depth of coverage along the genome. Using this idea and the result from Exercise 6, compute the expected fraction of the genome uncovered. What is the fraction of the genome covered by three or more reads? Convert these fractions into bp for the project.

Exercise 8. The function `o.lap` (which can be downloaded from <http://www.cmb.usc.edu>) constructs the portion of an overlap matrix resulting from comparing direct reads r_k and r_l with each other, recording any matches below the diagonal of the overlap matrix.

- Modify this function so that it will create the portion of the overlap matrix above the diagonal, corresponding to alignment of direct reads r_k to complementary reads r_l^* .
- The code fragment

```
#test for overlap of j with left end of i
s<-0 #Initialize counter
  for(q in 0:(m-1)){
    if(inseq[j,(k+q)]==le[q+1])
      {s<-s+1}
    ...
```

does not impose a penalty for mismatches when computing score s . Is a mismatch penalty needed in this application? Why or why not?

Exercise 9. Download the set of 40 sequence reads (8 nucleotides long each) in the file `r.reads.txt` from <http://www.cmb.usc.edu>. Perform the sequence assembly using `o.lap` and your modification of `o.lap` (Exercise 8) to create the overlap matrix. Use $m = 4$ and $t = 0.8$.

- For any sequence i , it is possible that sequences j that appear in the overlap matrix do not correctly overlap. How can you reduce this problem?
- Do you assemble a single contig? What does this indicate?
- The reads were not annotated with respect to the clones from which they came. If they had been, you might have detected mate pairs (paired-end reads). Suppose that you were told that reads 2 and 23* are from the same clone and 70 bp apart. How would this affect your sequence assembly?

Hints

The following code will be useful:

```
# Function for reverse complement of string:

r.star<-function(r){
#function to produce the reverse comp. of r.
#r is a string (vector) of length n
b<-c(4:1) #complements of 1,2,3,4
r.rev<-r[length(r[]):1] #reverses sequence
s<-r.rev
s[]<-b[r.rev[]] #replaces w/ complementary "letters"
return(s)
}
```

Syntax for extracting nonzero entries of overlap matrix (using sequence 9 as an example):

```
> (1:40)[tmp[9,]!=0]
[1] 2 3 7
```

All sequences that overlap with sequence 9

```
> c(((1:40)[tmp[9,]!=0]),((1:40)[tmp[,9]!=0]))
```

Output of overlapping sequences

```
r.reads[c(((1:40)[tmp[9,]!=0]),((1:40)[tmp[,9]!=0]),]
```

Exercise 10. This problem examines the effects of coverage and parameters on the assembly result.

- Sample `r.reads` without replacement, taking 25 of the sequences. Perform the assembly again as in Exercise 9 (with the same values of m and t). How does this affect the sizes of the assembled contigs and the number of islands?
- Using the full set of 40 sequence reads, what is the result of changing m from 4 to 5 with $t = 0.9$?

Signals in DNA

9.1 The Biological Problem

We consider a **signal** to be a DNA or RNA sequence pattern that is recognized by a protein or other molecule (or, for RNA, sometimes another region of the same molecule). Binding sites for proteins on DNA are important examples of signals, and in this chapter we focus on these signals. Particular instances of the sequence pattern may be represented by a sequence of letters in the pertinent alphabet (e.g., A, C, G, T for DNA), but for signals recognized by proteins, this is an approximation: proteins actually recognize specific atoms or groups of atoms on bases or base pairs. “Recognize” usually means binding in the thermodynamic sense. This may be strong binding, as in the case of repressors binding to operators, or binding may be transient, as with restriction endonucleases. Examples of common signals in DNA are:

- Restriction endonuclease recognition sequences (e.g., GAATTC for *EcoRI*).
- Binding sites for regulatory proteins: These proteins may function as repressors (e.g., *cI* protein of bacteriophage lambda, which binds to the corresponding operator), may regulate gene expression in response to physiological conditions of the cell (e.g., CRP protein, the cAMP receptor protein of *E. coli*), or may be eukaryotic transcription factors such as the glucocorticoid receptor (which binds to the GRE or glucocorticoid receptor element under appropriate conditions).
- Elements within replication origins and termination regions in genomes.
- Promoters: These are sites that determine where transcription is initiated. There are different classes of promoters, depending upon the type of RNA polymerase (eukaryotes) or type of specificity factor (prokaryotes) that acts at these sites. Within any particular class of promoters, individual examples have been fine-tuned by evolution to provide the appropriate level of transcription for the genes that they control. This means that promoter sequences display a much greater degree of sequence variability than

is observed for other signals (such as restriction endonuclease recognition sequences).

9.1.1 How Are Binding Sites on DNA Identified Experimentally?

Often the starting point is a cloned or PCR-amplified piece of DNA that is hypothesized to contain a binding site for a particular protein (e.g., a region near the 5' end of a gene, such as the *lac* promoter region). In most cases of interest, the DNA is duplex rather than single-strand. To determine experimentally whether a particular protein binds, it is also necessary to have purified (or partially purified) binding protein (e.g., CRP protein) stored in the freezer.

One way to identify whether a protein-binding site is present or absent on the DNA fragment is an **electrophoretic mobility-shift assay** (or gel-shift assay). In this experiment, duplex DNA in an appropriate buffer for protein binding is mixed with the binding protein and incubated to allow formation of the DNA-protein complex. The reaction mixture is then loaded onto a polyacrylamide gel, and electrophoresis is performed with naked DNA (no protein supplied) run in an adjacent lane. Binding of the protein to the DNA reduces the mobility of the DNA fragment relative to its mobility when there is no bound protein. The resulting shift in the position of the protein-bound fragment is visible after staining the gel. The amount of shifted complex depends upon how much protein was supplied and on its binding equilibrium constant.

Footprinting experiments can locate the positions of protein-binding sites on DNA molecules. In one implementation of this approach, duplex DNA containing the binding site is labeled at one end (on one strand) and is allowed to bind the protein, which covers a small segment of the DNA. The complex is then briefly treated with a nonspecific cleavage reagent such as DNase I, which cuts only in regions of the molecule *not* protected by the protein. Hydroxyl radicals can also be used for nonspecific cleavage. Each of the many DNA molecules in solution may be cleaved at one or more positions not covered by the protein. By electrophoresis of the treated DNA on a denaturing gel of the type used for analyzing DNA sequencing reaction products, single-strand fragments whose lengths extend from the labeled end to the various cleaved sites are resolved from each other, generating a “ladder” of bands having different lengths. The cleaved positions are located throughout the DNA *except for that portion of the DNA that was protected by the bound protein*. The region in the ladder where there are few (or fainter) bands is called the **footprint**. The location of the footprint is determined from a control “ladder” having fragments of known size.

Sometimes the investigator may possess proteins known to bind DNA but not instances of the cognate binding site in a DNA of interest. In that case, it is possible to retrieve fragments of DNA containing the binding site or sites by using **chromatin immunoprecipitation** (ChIP). (ChIP experiments are

used with other approaches, particularly genomic microarrays, to identify genomic locations of sites that bind particular proteins.) For example, if we have purified protein P known to bind to eukaryotic replication origins, it is possible to produce in rabbits (or rats, mice, or goats) specific antibodies that recognize protein P (see Section 1.5.2 for a more complete description). To identify DNA binding sites for P, we first perform *in vivo* chemical cross-linking of chromatin proteins, including P, to DNA. The chromatin is then extracted from the cells, and the DNA in the protein-DNA complexes is fragmented into smaller pieces (e.g., 500 bp long) by hydrodynamic shearing. Complexes of P with the DNA to which they are bound are then precipitated by using the anti-P antibodies (possibly with the aid of a secondary antibody). Cross-linking is reversed, and the DNA enriched for P-binding sites is purified. The sequence of the binding sites for P is revealed after cloning, sequencing, and alignment of the DNA. However, approximately half of the DNA recovered by ChIP does not result from specifically bound P. This is because the vast excess of proteins that bind weakly and nonspecifically nevertheless bind somewhat because of their higher concentrations, thus contributing to experimental “noise.”

9.1.2 How Do Proteins Recognize DNA?

We indicated that a signal is often a short stretch of a nucleic acid sequence to which one or more proteins may specifically bind. This may be represented by a string of letters. But what is the signal actually? After all, proteins cannot “read” Roman letters! Obviously, proteins are sensing chemical groups exposed on the DNA or RNA.

Proteins can recognize DNA or RNA by hydrogen bonding or electrostatic interactions with the phosphodiester backbone of the nucleic acid. Also, they may “read” the sequence of base pairs by hydrogen bond interactions with the donor and acceptor groups at the edges of the base pairs in the major or minor grooves. They also may employ hydrophobic interactions mediated by the methyl group of thymine or may partially intercalate a hydrophobic group between two base pairs (if the DNA is “kinked”). An example of site-specific recognition of DNA by the *engrailed* homeodomain protein is diagrammed in Fig. 9.1A. This illustrates interactions of specific amino acids with the major groove, the minor groove, and the phosphodiester backbone (shaded circles). What really counts for specific protein binding is the ability to recognize the actual sequence of base pairs, and particular patterns of hydrogen bond donor and acceptor groups on the edges of base pairs allow discrimination between them (Fig. 9.1B).

Protein-binding sites on DNA may be simple, as in the case of restriction endonuclease recognition sequences, or they may be complex. They may have internal symmetry (e.g., inverted repetition of sequence motifs). They may have highly conserved specific sequence patterns or may tolerate variation (degeneracy) at particular positions. DNA-binding proteins are commonly oligomers (e.g., dimers or tetramers) of smaller polypeptide chains.

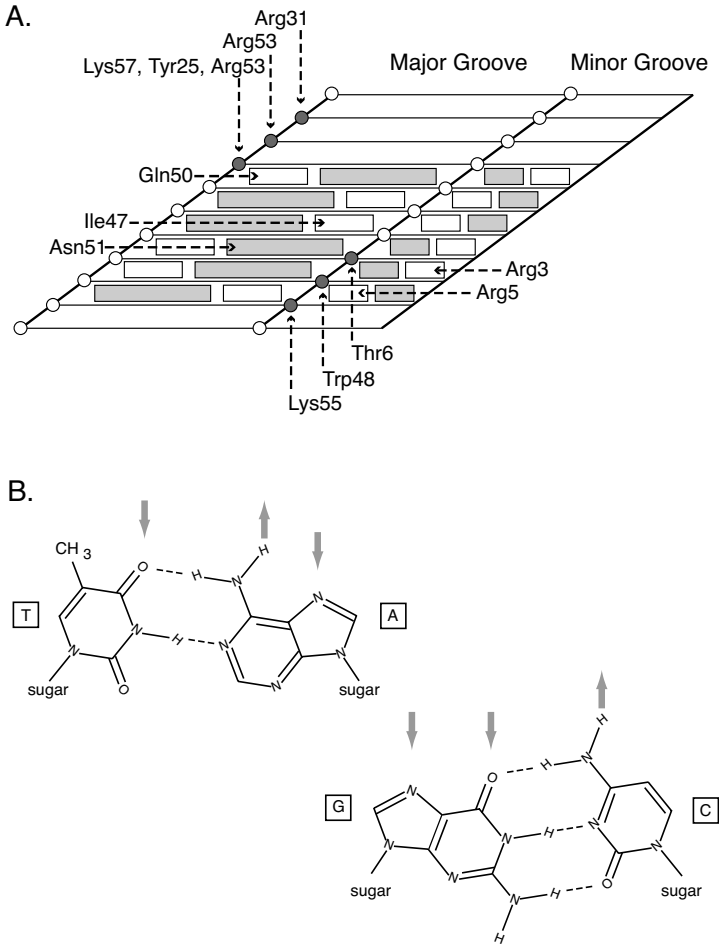


Fig. 9.1. How proteins interact with DNA. Panel A: Interactions between specific residues of the *engrailed* homeodomain protein with its binding site. The cylindrical surface of the DNA has been cut along one of the phosphodiester backbones and unwrapped for projection onto a plane. Bases A or T are represented by shaded or unshaded rectangles, respectively. Base pairs that do not specifically interact are not shown. Note that the bases indicated in the minor groove are the opposite edges of the same bases appearing in the major groove. Phosphate groups are represented by circles. Shaded circles represent phosphate groups with which *engrailed* protein specifically interacts. Dashed arrows indicate interactions of particular amino acid residues with specific bases or phosphate groups. Note that there are interactions between the protein and the bases in both the major and the minor grooves. Underlying data were taken from Kissinger CR et al. (1990) *Cell* 63:579–590. Panel B: Locations of hydrogen bond donor groups (arrows pointing away from the structure) and acceptor groups (arrows pointing toward the structure) on the edges of the Watson-Crick base pairs in the major groove. Particular sequences create characteristic patterns of donor and acceptor groups, particularly in the major groove.

Each polypeptide may be capable of binding DNA on its own, but as an oligomer, the binding strength is increased. If the oligomer is a dimer having a “head-to-head” arrangement, then the binding protein will have twofold rotational symmetry, and therefore the site on the DNA to which it binds will also have twofold rotational symmetry, recognized as inverted repetition of the DNA sequence (Fig. 9.2). An example of what is meant by inverted repetition is the GRE (glucocorticoid receptor element):



The underlined bases appear in an opposite direction relative to the laboratory frame of reference, hence the “inverted” adjective. These repeated bases are on opposite strands. The two symmetrically disposed strings of conserved bases in a binding site are called *half-sites*. Notice that, in this case, the binding site can be recognized by examining either strand of the DNA for **AGAACA**. For sites that lack this symmetry, it is necessary either to scan each strand separately to find the binding sites on a duplex DNA or scan one strand for the two strings representing both the site *and* its complement, thus effectively examining both the strand and its complement in a single pass.

Figure 9.2 illustrates the relationship between binding protein and binding site structure using bacteriophage lambda *cro* protein as a specific example. *cro* protein (Cro) can bind to lambda operator sites. Cro is a homodimer having twofold rotational symmetry. Therefore, the operator also has twofold rotational symmetry (i.e., it has inverted repeat structure). Moreover, because each *cro* protein monomer binds by using an alpha helical region that is inserted into the major groove, interaction between successive major groove regions on the same side of the DNA helix places the centers of the half-sites approximately 10 bp away from each other. (The average helix pitch of B-form DNA is 10.4 bp.)

The sites recognized by *cro* protein are summarized in Table 9.1. Observe that the sequences of the six operator sites are similar, but not identical, and that the sequences of half-sites comprising the same operator are similar, but not identical. Obviously, any realistic description of signals in DNA must be capable of representing this sequence variation.

9.1.3 Identifying Signals in Nucleic Acid Sequences

Given a set of DNA or RNA sequence strings, how can we determine what signals they encode, and how can we represent these signals? One approach seeks to gain knowledge of a pattern that has not been previously specified (an **unsupervised** approach). This is a daunting, and in strictest terms impossible problem. However, we know a good deal about the length and extent of protein-binding patterns in DNA, as illustrated in the previous section. An example of unsupervised pattern discovery is identifying *k*-words that are over-represented in a set of functional sequences such as promoters. We provided

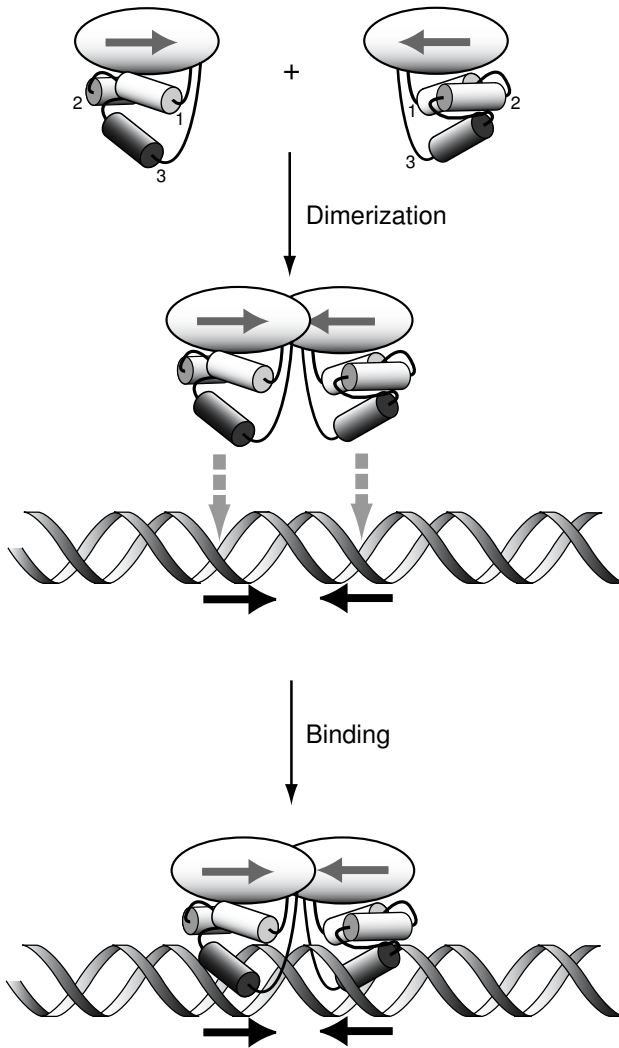


Fig. 9.2. How site structure reflects the structure of a binding protein. The binding protein (Cro in this case) binds head-to-head as a dimer. The individual monomer components interact with the DNA through the major groove (bottom panel). Interaction of the dimer with successive major groove regions on the same face of the helix implies that the centers of the two interaction regions will be spaced 10 to 11 bp apart—the pitch of B-form DNA. The equivalence of interactions between helices 3 of both monomers and DNA imposes inverted symmetry within the DNA sequence to which Cro binds.

Table 9.1. Binding sites for lambda *cro* protein at operators in the *cI* region of bacteriophage lambda (Gussin et al., 1983). Underlined base pairs separate each site into two half-sites.

O _L 1	5'-T A T C A C C G <u>C</u> C A G T G G T A-3'
	3'-A T A G T G G C <u>G</u> G T C A C C A T-5'
O _R 1	5'-T A T C A C C G <u>C</u> C A G A G G T A-3'
	3'-A T A G T G G C <u>G</u> G T C T C C A T-5'
O _L 2	5'-T A T C T C T G <u>G</u> C G G T G T T G-3'
	3'-A T A G A G A C <u>C</u> G C C A C A A C-5'
O _R 2	5'-T A A C A C C G <u>T</u> G C G T G T A T G-3'
	3'-A T T G T G G C <u>A</u> C G C A C A A C-5'
O _L 3	5'-T A T C A C C G <u>C</u> A G A T G G T T-3'
	3'-A T A G T G G C <u>G</u> T C T A C C A A-5'
O _R 3	5'-T A T C A C C G <u>C</u> A A G G G A T A-3'
	3'-A T A G T G G C <u>G</u> T T C C C T A T-5'

an extensive illustration of this in Section 3.6. In that case, we expected there to be signals in the regions immediately upstream of the 5' end of transcribed gene regions, but we did not specify what they were. Also, we did not require that the signals be present in all members of the set. We were able to identify over-represented k -words such as TAAT and ATAA that are contained within TATAAT (−10 region) without alignment or experimental specification of the signal. Unsupervised methods have been used to identify k -words representing both known and previously unrecognized regulatory sequences for yeast genes (Brazma et al., 1998; van Helden et al., 1998). A statistical approach devised by Bussemaker et al. (2000) can be used to generate a k -word dictionary of regulatory sites or other significant motifs. Unsupervised methods are not discussed further here, but see Section 14.4.3.

Another approach, **supervised** learning, includes prior knowledge of the pattern to be found. For example, multiple instances of a known and aligned DNA signal can be used to define parameters of probabilistic models describing that signal. This is the subject of the remainder of this chapter.

9.2 Representing Signals in DNA: Independent Positions

There are several ways of representing signals in DNA, ranging from very simple to rather complex. The simplest method of representing a binding site is as a **consensus sequence**—a string of characters corresponding to the most common occurrences of bases at each position. This is adequate for sites such as restriction endonuclease recognition sequences and reasonably good for sites such as GRE half-sites (examples aligned below).

```

Position:  123456
           AGAACA
           ACAACA
           AGAACA
           AGAAGA
           AGAACA
           AGAACT
           AGAACA
Consensus: AGAACA

```

In this particular collection of sites, one of the four bases is much more likely to occur at any position in the site than any of the three others. (Exceptions to the consensus base at any position are underlined.) The consensus sequence is a listing of the preferred bases at each position. If there are positions at which either of two bases is commonly found, then the consensus might be written to indicate both possibilities. So, for the example above, we might write $A(C/G)AA(C/G)(A/T)$ if either of the indicated letters is tolerated at positions 2, 5, and 6. There are cases, however, where there is much more sequence variation at some or all of the positions within the binding site. For such cases, consensus methods are not appropriate and a probabilistic description is required.

A string of iid letters does not contain signals, except by chance. To generate a string of iid letters, a base is assigned at each position according to a probability distribution, which might be represented as a column vector (4×1 matrix) whose entries correspond to the probabilities of A, C, G, or T determined by the overall base composition of the DNA. We described this process in Chapter 2. In the example of the GRE half-site above, it should be obvious that there is no single probability distribution for describing the site. Instead, we must specify (in principle) a different distribution at every single position (in other words, we are removing the “id” from “iid”). This suggests another representation of protein-binding sites. We create a matrix of probabilities by “binding” together (the `cbind` function in R) the column vectors that represent the probability distributions (probabilities of each letter for any position) at each position in a site. Therefore, each element in this matrix represents the probability of finding a particular base at a given position.

For example, the probability distribution at position 2 for the limited sample of GRE half-sites listed above, with rows corresponding to A, C, G, and T, respectively, is

$$\begin{bmatrix} 0.00 \\ 0.14 \\ 0.86 \\ 0.00 \end{bmatrix}$$

and the matrix representing all six positions in the half-site is

$$\begin{bmatrix} 1.00 & 0.00 & 1.00 & 1.00 & 0.00 & 0.86 \\ 0.00 & 0.14 & 0.00 & 0.00 & 0.86 & 0.00 \\ 0.00 & 0.86 & 0.00 & 0.00 & 0.14 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.14 \end{bmatrix}$$

where rows $i = 1, \dots, 4$ correspond to **A**, **C**, **G**, and **T**, respectively, and columns $j = 1, \dots, 6$ correspond to the positions within the site. This representation of the sites is called a **positional weight matrix**, or **PWM**. (See Stormo, 2000a for a review of the history of PWMs.) Note that the six lambda operator half-sites Table 9.1 display more sequence variation at each position than is seen at any position in the GRE half-sites, and thus the positional weight matrix provides a much more precise representation of these half-sites than would a consensus sequence. There are other cases where consensus methods are very imprecise and inappropriate, such as for promoter sequences.

In preparation for describing and analyzing a signal in DNA, we gather together *and align* representatives of that signal (see below). A **training set** is a collection of bona fide signals (or sites) used to produce the probabilistic model. A second set of bona fide signals or sites is necessary for testing the mathematical description or model. This is called the **validation set**. Sometimes the construction of the model requires a **challenge set** of sequences that are not sites. The challenge set is the one to which the real sequences in the training set are contrasted, such as a set of “nonsites” to be contrasted with sites. In some circumstances, we might use a probabilistic model to produce the challenge set (for example, an iid model or a Markov chain model). The process of estimating the probability distributions from the training set is called *learning* in the machine-learning world.

9.2.1 Probabilistic Framework

Our practical goal is to recognize a signal of length w within a string representing a DNA sequence. To do this, we parse the sequence using windows of width w and ask how well the w letters in that window correspond to a particular signal. This means that we need to assign some type of score to the sequence in each successive window. We assume that we have a collection of aligned DNA sequences of the same length w and having no gaps, and that we know that all members of this collection are sites for binding a particular DNA-binding protein. How do we assign a score to any particular sequence to describe its level of correspondence with the training set?

We follow the logical steps presented in Durbin et al. (1998). Given a collection of aligned sites, each of length w , let p_{ai} be the probability of any particular letter a from $\{\mathbf{A}, \mathbf{C}, \mathbf{G}, \mathbf{T}\}$ at position i measured within a site. Identities of letters at each position in the sequence are assumed to be independent of identities of their neighbors. (Remember that since we are talking about *signals*, the probability distributions for the letters at each position are *not* identical in the general case.)

The probability of a sequence $\mathbf{A} = a_1 a_2 \dots a_w$ given hypothesis \mathcal{B} that it is a binding site is

$$\mathbb{P}(\mathbf{A}|\mathcal{B}) = \prod_{i=1}^w p_{ai}. \quad (9.1)$$

Suppose that the same sequence was chosen from a random collection of bases (hypothesis \mathcal{R}) with probabilities q_{ai} . (The random model is iid.) Then the probability of the sequence is given by

$$\mathbb{P}(\mathbf{A}|\mathcal{R}) = \prod_{i=1}^w q_{ai}, \quad (9.2)$$

and the odds ratio is given by

$$\frac{\mathbb{P}(\mathbf{A}|\mathcal{B})}{\mathbb{P}(\mathbf{A}|\mathcal{R})} = \prod_{i=1}^w \frac{p_{ai}}{q_{ai}}. \quad (9.3)$$

The *score* for the sequence \mathbf{A} can be defined as the log of the odds ratio, or

$$S = \log_2 \frac{\mathbb{P}(\mathbf{A}|\mathcal{B})}{\mathbb{P}(\mathbf{A}|\mathcal{R})} = \sum_{i=1}^w \log_2(p_{ai}/q_{ai}) \equiv \sum_{i=1}^w s_i. \quad (9.4)$$

where $s_i = \log_2(p_{ai}/q_{ai})$ is the contribution to the sequence score of the base at position i . (Note that we are taking the log to base 2 because eventually we will be referring to “information” in bits. Note that $\log_2(x) = \ln(x)/\ln(2)$.)

The p_{ai} may be used as elements in a positional weight matrix. For the contrasting iid model, the q_{ai} are independent of position (identically distributed), so from here on, we just refer to q_{ai} as q_a . If the base composition is 50% G+C, then $q_a = 0.25$ for each base for duplex DNA (i.e., $q_a = 0.25$ for all a in $\{\mathbf{A}, \mathbf{C}, \mathbf{G}, \mathbf{T}\}$).

Let's consider a more complex example than those we have considered previously. *Escherichia coli* promoters have been recognized to contain conserved hexamer sequences at approximately -10 and -35 relative to the first position in the transcript. The consensus for the -10 hexamer is often represented as TATAAT. Table 9.2 lists nine examples of promoter segments for *E. coli*; these are part of a larger data set given in Appendix C.3. Clearly, it is hard to pick out the -10 hexamer in all cases using the consensus given above, so we represent the -10 hexamer as a positional weight matrix using compiled data for a subset of all *E. coli* promoters (Harley and Reynolds, 1987; Stormo, 1990). First, we create a matrix containing values for counts of each letter at each position (rows $i = 1, \dots, 4$ corresponding to A, C, G, and T, and columns corresponding to positions as before):

$$\begin{bmatrix} 9 & 214 & 63 & 142 & 118 & 8 \\ 22 & 7 & 26 & 31 & 52 & 13 \\ 18 & 2 & 29 & 38 & 28 & 5 \\ 193 & 19 & 124 & 31 & 43 & 216 \end{bmatrix}$$

Table 9.2. A sample of *E. coli* promoter sequences. These sequences have been aligned relative to the transcriptional start site at position +1 (boldface large letter). Sequences from -40 to +11 are shown. Close matches to consensus -35 and -10 hexamers are underlined. See also Appendix C.3 for additional examples and sources of the data.

	-35		-10		-1
ORF83P1		CTCTGCTGGC <u>ATTCACA</u> AAATGCGCAGGGG <u>TAAAAC</u> GTTTCCTGTAGCACCG			
<i>ada</i>		GTTGGTTTTTGC	GATGGTGACCGGCAGCCTAAAGGCTA	TCCTTAACCA	
<i>ammP4</i>		TTCACATTTCTGT	<u>GACA</u> TACTATCGGATGTGCGGTAATTG	TATGGAACAGG	
<i>araFGH</i>		CTCTCCTATGGAGAATTAATTTCTCG	<u>CTAAAA</u> CTATGTCAA	CACAGTCACT	
<i>aroG</i>		CCCCGTTTACACATTCTGACGGAAGATATAGATT	GGAAGT	A	TGCATTAC
<i>atpI</i>		TATTGTTT	<u>GAAA</u> TCACGGGGCGCACCGT	<u>TATAA</u> T	TGACCGCTTTTTGATG
<i>caiT</i>		AATCACAGAATACAGCTTATTGAATACCC	<u>ATTAT</u> GAGTTAG	CCATTAACGC	
<i>clpAP1</i>		TTATTGACGTGTTACAAAAATCTTTTCTT	<u>TATGAT</u> GTAGAA	A	CGTGCAACGC
<i>crrP2-I</i>		GTGGTGAGCTTGCTGGCGATGAACGTGCT	<u>TACACT</u> TCTGTT	GCTGGGGATGG	

The sums of the entries in each column are all 242, which is the number of promoter sequences in this particular data set. The numerical entries indicate an overwhelming preference for A at position 2 and a less pronounced preference for A at position 5.

We can also represent the data by a matrix containing the relative frequencies of each letter at each position:

$$\begin{bmatrix} 0.04 & 0.88 & 0.26 & 0.59 & 0.49 & 0.03 \\ 0.09 & 0.03 & 0.11 & 0.13 & 0.21 & 0.05 \\ 0.07 & 0.01 & 0.12 & 0.16 & 0.12 & 0.02 \\ 0.80 & 0.08 & 0.51 & 0.13 & 0.18 & 0.89 \end{bmatrix}$$

Entries in this matrix correspond to entries in the previous one, each divided by 242. Note that the sums of the entries in each column should be 1.0, and they are, to within rounding error.

We can also represent the data by a third matrix (used for scoring), containing the corresponding values for $\log_2(p_{ai}/q_a)$. Such a matrix is known as a **position-specific scoring matrix** (PSSM—articulated as “possom”). If we make the reasonable approximation for *E. coli* DNA that $p_a = 0.25$ for all a , the resulting PSSM is:

$$\begin{bmatrix} -2.75 & \boxed{1.82} & 0.06 & \boxed{1.23} & \boxed{0.96} & -2.92 \\ -1.46 & -3.11 & -1.22 & -0.96 & -0.22 & -2.22 \\ -1.75 & -4.92 & -1.06 & -0.67 & -1.11 & -3.60 \\ \boxed{1.67} & -1.67 & \boxed{1.04} & -0.96 & -0.49 & \boxed{1.84} \end{bmatrix}$$

Now let's see how to score a particular sequence string. Take the sequence ACTATAATCG for example. If we start parsing this from the beginning using windows of width 6, we score hexamers ACTATA, CTATAA, TATAAT, ... until we come to the end. The scores s_i at each position for TATAAT (matches with the consensus) are boxed in the matrix above. The score for TATAAT is $S = 1.67 + 1.82 + 1.04 + 1.23 + 0.96 + 1.84 = 8.56$. From (9.4), it is clear that $\mathbb{P}(\mathbf{A}|\mathcal{B})/\mathbb{P}(\mathbf{A}|\mathcal{R}) = 2^S$, so for TATAAT, $\mathbb{P}(\mathbf{A}|\mathcal{B})/\mathbb{P}(\mathbf{A}|\mathcal{R}) = 377$. In contrast, for ACTATA, $S = -2.75 - 3.11 + 1.04 + 1.23 - 0.49 - 2.92 = -9.75$.

Suppose that only one base occurred at each position in every example in the training set (e.g., only T appeared at position 1, only A at position 2, etc.). Then $\log_2(p_{ai}/q_a)$ would have been $\log_2(1/0.25) = \log_2 4 = 2$ at each position. The score for the six positions would then have been 12, and $\mathbb{P}(\mathbf{A}|\mathcal{B})/\mathbb{P}(\mathbf{A}|\mathcal{R}) = 2^{12} = 4096$. This is the same ratio of probabilities that was implicit when we were looking for an invariant 6 bp endonuclease recognition site in DNA (see Section 3.2.2).

A positional weight matrix is a probabilistic description of a protein-binding site or other signal, with the underlying assumption that the state or letter at a given position is unaffected by the state or letter at the previous position. We have emphasized the utility of such matrices for scoring potential sites within a string containing other DNA. In addition, a PWM (elements representing probabilities at each position) is a probabilistic model that allows us to simulate sites, should we wish to do so. When we wanted to simulate a string of iid letters (see Chapter 2), we used a single vector of probabilities to generate a letter for each site in a particular simulated sequence. To simulate a binding site of length w , we use consecutively w probability vectors, each corresponding to a column in the PWM. This is done in one of the exercises at the end of this chapter.

9.2.2 Practical Issues

We briefly touch on some practical matters that we have ignored up to now. The first is aligning the sequences. For *E. coli* promoters, this was simplified because the alignment could be approximately fixed relative to the transcriptional start site. (The +1 position is determined experimentally.) For short patterns or signals embedded in extensive other DNA, alignment is a more difficult proposition (i.e., finding statistically meaningful short local alignments in long sequences). Having footprinting data helps to restrict the lengths of the sequences to be aligned.

A second problem is defining the extent of the site (the value of w). Here, footprinting data are also extremely helpful. The **information** content $I(X_i)$,

mentioned in Section 9.4, can be used to determine the size of the sites. The approach is to calculate $I(X_i)$ across the sequence containing the site and to select those positions where $I(X_i)$ is significantly above the background level.

A third problem is that only a limited sample of the total population of sites is available for use as a training set. This means that the elements p_{ai} of the positional weight matrix are only estimates of the population values. To correct for a small sample size, we may compute p_{ai} as

$$p_{ai} = \frac{n_{ai} + 1}{N + 4}, \quad (9.5)$$

where n_{ai} is the number of sites having letter a at position i , and N is the total number of sites in the training set. This is particularly necessary when $n_{ai} = 0$ since the $\log_2(p_{ai}/q_a)$ term would then be undefined. This equation provides a correction for small sample sizes by adding a “pseudocount” of 1 for each base at each position. The 4 in the denominator reflects the fact that a pseudocount of 1 has been supplied for each of $\{A,C,G,T\}$. If there were no observations at all ($n_{ai} = N = 0$), the result would be $p_{ai} = 1/4$, which is what we would have predicted with no prior knowledge of the sites (assuming equal probabilities of each base at each site).

Finally, note that, for a signal of length w , a positional weight matrix contains $4w$ parameters. For a matrix of probabilities, only three entries in each column are independent since the sum of the four entries must equal 1.0. Therefore, only $3w$ parameters are independent. If there were only six members in the training set, there would on average be only two observations available for estimating each parameter ($6w$ observations $\div 3w$ independent parameters). This is such a small sample size that we would not expect the parameters to be reliably estimated. The number of bona fide sites available depends on wet-lab experiments, and this number may not be large enough to provide both training and validation sets of adequate size. Sometimes methods such as cross-validation (leave one out) are employed to test the predictive performance of PWMs when the number of sites is too limited to provide both training and validation sets of adequate size.

In Computational Example 9.1, we reinforce these concepts using the site to which transcription factor GATA-1 binds. GATA-1 is a transcription factor that regulates transcription in hematopoietic cells (cells that give rise to blood cells, such as red blood cells or erythrocytes). The string representing its binding site has $w = 6$, and it is represented by the consensus (A/T)GATA(A/G). Examples of GATA-1 binding sites are listed in the TRANSFAC database (<http://transfac.gbf.de/TRANSFAC/>). (The database may list the binding site embedded in a string of other sequence or may list either of two strands. We need to perform alignment and record the complements of the sequence given, as appropriate.) We chose GATA-1 binding sites for this example because there is a reasonably large number of listed binding sites for estimating the parameters (49 human sites). These site sequences are listed in Table 9.3.

Table 9.3. Binding sites for hematopoietic transcription factor GATA-1 from *H. sapiens*. Source: TRANSFAC database (<http://transfac.gbf.de/TRANSFAC/>).

TTATAG	AGATAA	TGATTA
AGATAT	TGATAA	AGATAA
AGATAG	AGATAA	AGATAG
ATATCT	AGATAG	TGATAT
AGATAG	TGATAG	AGATAA
AGATAG	TGATCA	TCAGAG
AGATAG	TTATCA	AAGTAG
TGATAA	AGATGG	AGATTA
AGATAA	TGATAT	TGATAG
AGATAA	AGATAG	TGATAG
CGATAG	TGATAA	AGATAC
AGAGTT	GGATAC	TGATTG
TGATAA	AGATAA	AGATTA
TGATAA	CGATAA	AGAATA
AGATGG	TGATAG	AGATAA
AGATAG	AGATAA	AGATTA
AGATTG		

Computational Example 9.1: PWM representation of GATA-1 sites

We begin by converting the data in Table 9.3 into a numerical representation, as we have done in earlier chapters. The data are stored in the matrix `gata`. The first three entries are:

```
> gata
      [,1] [,2] [,3] [,4] [,5] [,6]
[1,]    4    4    1    4    1    3
[2,]    1    3    1    4    1    4
[3,]    1    3    1    4    1    3
```

We can produce a PWM in terms of probabilities of bases at each site directly from `gata`. However, we ultimately want to generate scores according to (9.4), so we also create a PSSM having elements of the form $\log_2(p_{ai}/q_a)$. This means that we need the probabilities of each base for genomic DNA. Since these sites are from human DNA, we use the human base composition, 41% G+C, to produce the “background” probability distribution based upon the iid model. Vector `bg` has four elements corresponding to the probabilities of A, C, G, and T, respectively:

```
> bg<-c(0.295,0.205,0.205,0.295)
```

```
> bg
[1] 0.295 0.205 0.205 0.295
```

Now we write a function to compute the PWM, given a matrix of sites and the background probability distribution as input:

```
makepwm<-function(x,bg){
# x = matrix of N aligned sites coded numerically
# bg = vector (1x4) of background base frequencies
L<-length(x[1,]) # Number of positions in each site
N<-length(x[,1]) # Number of sites
pwm<-matrix(rep(1,4*L),nrow=4)
# pwm initialized to 1 for each matrix element (pseudocounts)
for (j in 1:L){
  for (i in 1:N){
    k <- x[i,j]
    pwm[k,j] <- pwm[k,j]+1
  }
}
N <- N+4 # Denominator for small sample correction
pwm<-pwm/N # PWM in terms of probabilities
log2pwm<-matrix(rep(0,4*L),nrow=4,ncol=L)
# Initialize PWM in terms of log(base 2) of p/q
for(i in 1:4){
  log2pwm[i,]<-log2(pwm[i,]/bg[i])
  # Scores for each [nucleotide, position], base 2
}
return(pwm, log2pwm)
}
```

The “guts” of this function are the lines

```
for (i in 1:N){
  k <- x[i,j]
  pwm[k,j] <- pwm[k,j]+1
}
```

Here we have used the numeric coding of the sequence to identify k , which in the second line after the `for` also specifies the numeric row number corresponding to each base in object `pwm`. The vectorization in R avoids a set of commands such as

```
if(x[i,j]==1) ...pwm[1,j]<-pwm[1,j]+1 ...elseif(x[i,j]==2) ...
```

Notice that we are returning two objects from this function. This produces an R “list” object, which we call `tmp`. Objects in the list `tmp` are extracted by entering `tmp` followed immediately by the object name prefixed with the `$` sign:

```

> tmp<-makepwm(gata,bg)
> tmp$pwm
      [,1]      [,2]      [,3]      [,4]      [,5]      [,6]
[1,] 0.566040 0.037736 0.924530 0.037736 0.698110 0.433960
[2,] 0.056604 0.037736 0.018868 0.018868 0.075472 0.056604
[3,] 0.037736 0.849060 0.037736 0.056604 0.056604 0.396230
[4,] 0.339620 0.075472 0.018868 0.886790 0.169810 0.113210

> tmp$log2pwm
      [,1]      [,2]      [,3]      [,4]      [,5]      [,6]
[1,] 0.94018 -2.9667  1.6480 -2.9667  1.24270 0.55685
[2,] -1.85670 -2.4416 -3.4416 -3.4416 -1.44160 -1.85670
[3,] -2.44160  2.0502 -2.4416 -1.8567 -1.85670  0.95070
[4,] 0.20322 -1.9667 -3.9667  1.5879 -0.79678 -1.38170

```

(You should check for yourself to see what happens if you enter object name `tmp` alone.)

Now we use `log2pwm` to generate scores for all of the input sites in `gata`. To do this, we need a scoring function, which is shown below. This calculates the score for the first L positions of any input sequence, where L is the site length, here represented as `length(log2pwm[1,])`. We invoke this repeatedly for the set of GATA-1 sites, and later we use it in another function to scan along a long input sequence.

```

calcscore<-function(seq,log2pwm){
# seq is a vector representing input DNA numerically
# log2pwm is a PWM (4xL) with elements as log base 2
score <- 0
for (j in 1:length(log2pwm[1,])){
  score<-score+log2pwm[seq[j],j]}
return(score)
}

```

Now we calculate scores for all of the GATA-1 sites. First, we extract `log2pwm` from `tmp` as a separate object:

```
> log2pwm<-tmp$log2pwm
```

Then we loop `calcscore` over all elements in `gata` and store the resulting scores in `gata.score`:

```

> gata.score<-rep(0,length(gata[,1]))
> for(i in 1:length(gata[,1])){
+ gata.score[i]<-calcscore(gata[i,],log2pwm)
+ }
>

```

```
> signif(gata.score,2)
[1] 3.67 6.09 8.42 -0.61 8.42 8.42 8.42 7.29 8.03 8.03
[11] 5.62 0.60 7.29 7.29 5.32 8.42 6.38 8.03 7.29 8.03
[21] 8.42 7.68 4.60 0.59 5.32 5.35 8.42 7.29 2.23 8.03
[31] 5.23 7.68 8.03 5.25 8.03 8.42 5.35 8.03 -0.25 -0.69
[41] 5.99 7.68 7.68 5.61 5.64 5.99 1.43 8.03 5.99
```

(We use the `signif()` function to limit the output to two significant figures). We examine the score distribution graphically:

```
> hist(gata.score,xlim=c(-12,12),ylim=c(0,0.4),
+ nclass=10, prob=T)
```

The result is shown in Fig. 9.3. The score distribution is very broad and asymmetric. Notice that some of the GATA-1 sites have scores below 0. What would you have expected the average score to be for a set of *background* sequences composed of iid letters? We return to this question in the example after the last section of this chapter. We might wonder whether some of the GATA-1 sequences in our training set have been misidentified.

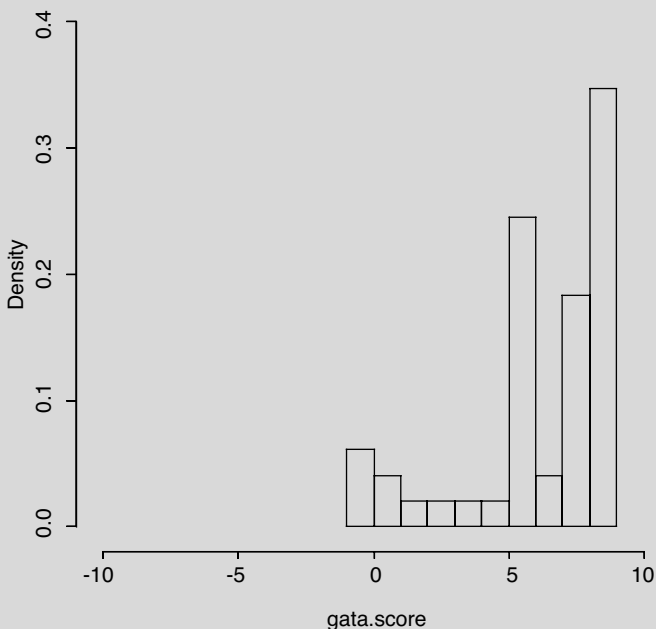


Fig. 9.3. Histogram of GATA-1 site scores for sites listed in Table 9.3.

9.3 Representing Signals in DNA: Markov Chains

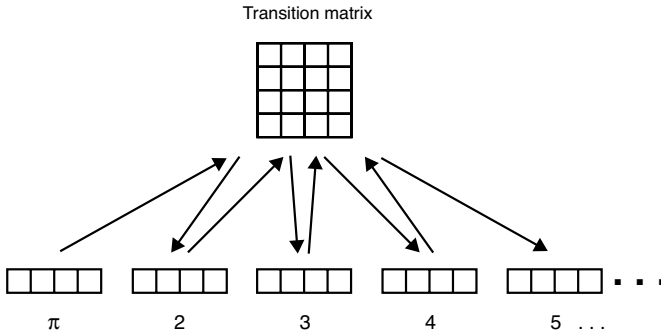
A PWM represents a signal under the assumption that the identities of letters at a given position are independent of the identities of letters at the previous position. The probability distributions in general differ at every position in the site (i.e., the letters are not identically distributed). We can remove the condition of independence by using a Markov chain description. We have already used this in Chapter 2 to describe DNA sequences having dinucleotide frequencies that departed from the iid predictions. That application assumed that identities of bases at various positions were not independent but *were* identically distributed at all positions (i.e., the same transition matrix was employed at every position).

The comparison and contrast of the approach in Chapter 2 with the one used here is illustrated in Fig. 9.4. When we were representing a string with identical distributions of each letter, we started with an initial probability distribution π and produced the probability distributions at each subsequent position by multiplication using a single transition matrix, as diagrammed in Fig. 9.4A. Only one transition matrix was required because we were considering *homogeneous* chains. The Markov chain illustrated in the diagram in Fig. 9.4A is conventionally represented as shown in Fig. 9.5. To represent the signals, the π vector is identified with the vector of probabilities of each state at the first position, and subsequent probabilities are produced by multiplication by a succession of $(w-1)$ transition matrices (Fig. 9.4B). Multiplication of the vector of first position probabilities by matrix 2 produces the probability distribution at position 2 given the distribution at position 1, multiplication of the probability distribution at position 2 by matrix 3 produces the probability distribution at position 3 given the distribution at position 2, and so on. It should be evident to you that the initial vector and the set of transition matrices are sufficient to allow simulation of the binding site type to which they correspond.

It is important to think about the *data structure* and the number of parameters. For PWMs, the probabilistic data were summarized in a matrix that could be produced by binding together column vectors corresponding to the independent probability distribution at each site. For the Markov chain representation, we can stack the individual transition matrices to form a three-dimensional *array*, or three-dimensional matrix. The R statistics package accommodates this type of data construct. With the Markov chain description, the number of parameters has increased compared with the positional weight matrix representation. Remember that each row of a transition matrix consists of the probabilities that the next letter is A, C, G, or T, given the letter corresponding to the row label. Three of these probabilities are independent, so $4 \times 3 = 12$ of the entries in each transition matrix are independent. There are three independent probabilities in the vector for position 1, and there are $(w-1)$ transition matrices. Therefore, the number of parameters in this prob-

A.

Markov chain, identical distributions



B.

Markov chain, nonidentical distributions

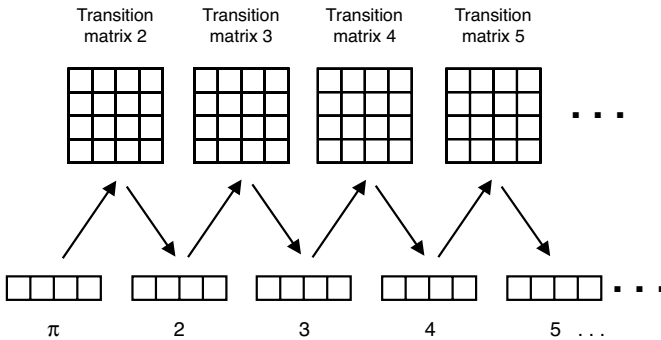


Fig. 9.4. Contrast between the Markov chain representation for positions with identical probability distributions (panel A) and the model for positions in a signal (binding site) sequence (panel B). Numbered four-element horizontal rows are the vectors of probabilities for A, C, G, and T at each position. These vectors are transformed to vectors at the next position by matrix multiplication by a transition matrix. In panel A, a single transition matrix is employed. In panel B, a different transition matrix is required for each successive position.

abilistic description is $3 + 12(w - 1)$, or for $w = 6$ a total of 63 parameters (in contrast with 18 parameters for a conventional PWM representation, $w = 6$).

In Computational Example 9.2, we represent the GATA-1 sites by using a Markov chain.

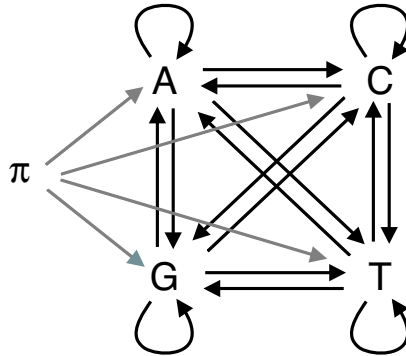


Fig. 9.5. A graphical representation of the Markov chain model for DNA that does not contain a signal. Arrows indicate allowable transitions from one state to another, and associated with each arrow is a probability for that transition. The initial state is drawn from the initial probability distribution π . In any chain generated by the model, state $n + 1$ is assigned based upon the identity of the state at n and the probabilities describing the model. The 16 dark arrows correspond to the elements in the transition matrix and the lighter arrows to the entries in the initial probability distribution, Fig. 9.4A. (Redrawn from Durbin et al., 1998.)

Computational Example 9.2: Markov chain representation of GATA-1 sites

Step 1: Preprocess the data

We find it convenient to use logical operators, so we first convert the alphabetical characters into numeric values: A= 1, C= 2, G= 3, and T= 4. A space is placed between each character. This can be done with Perl or with any text editor. The data may be put in a file called `gata1N.txt`.

Step 2: Import the data

Read the data into a matrix object defined under R. If you do not use a UNIX text editor, you may find that formatting elements are picked up by `read.table`. Save your file as *text only* before using `read.table`.

```
> gata<-read.table("gata1N.txt", header=F)
```

Step 3: Make the probability vectors

Produce the vector representing the probability distribution at position 1. This can be done by simple base counts and using the correction for small sample sizes given in the last section.

```

> length(gata[gata[,1]==1,1])
[1] 29
> length(gata[gata[,1]==2,1])
[1] 2
> length(gata[gata[,1]==3,1])
[1] 1
> length(gata[gata[,1]==4,1])
[1] 17
> vectorn<-c(29,2,1,17)
> vectorn
[1] 29 2 1 17

```

Now, applying the small sample correction, we estimate the probabilities from the frequencies and check that the probabilities sum to unity, as they should:

```

> vector1[]<-(vectorn[]+1)/(49+4)
> vector1
[1] 0.56603774 0.05660377 0.03773585 0.33962264
> sum(vector1[])
[1] 1

```

`vector1` represents the probability distribution for the first position of this collection of GATA-1 binding sites.

Step 4: Produce the transition matrices

We indicate how this is done for `matrix2` (used to create the probability distribution of sites at position 2 given `vector1`). In general, we need to read down each column of `gata` for columns 2 through 6, evaluating the number of times that 1, 2, 3, or 4 appears in a column (except the first) given that each instance is preceded by 1, 2, 3, or 4 in the previous column. We first do this by hand, to see how it works, and then by using an R function. (You may wish to revisit Section 2.6, where we used R to implement a first-order Markov process that generated a sequence having predetermined dinucleotide frequencies.)

[a.] Initialize a matrix to hold counts, `matrix0`.

```

> matrix0<-matrix(nrow=4,ncol=4,rep(0,16))
> matrix0
      [,1] [,2] [,3] [,4]
[1,]    0    0    0    0
[2,]    0    0    0    0
[3,]    0    0    0    0
[4,]    0    0    0    0

```

[b.] Record the following numerical counts in `matrixn` (4×4): `matrixn[1,1]` is the number of times that a 1 in the first column of the `gata` matrix is followed by a 1 in the second column for the whole list of binding sites. By inspection of Table 9.3 (in numerical form), we see that this happens one time (row 40 of `gata`), so the `[1,1]` element is 1.

`matrixn[1,2]` is the number of times that a 1 in the first column of the `gata` matrix is followed by a 2 in the second column. By inspection of Table 9.3, we see that there are no instances of this at all, so we record 0 for the `[1,2]` element of `matrixn`.

We do one more entry by hand. `matrixn[4,3]` is the number of times that a 4 in the first column of the `gata` matrix is followed by a 3 in the second column. There are 14 occurrences of 4 followed by 3, so we record 14 in `matrixn` element `[4,3]`.

Once we have calculated all of the matrix elements in `matrixn`, we need to convert the numerical counts to probabilities to create the transition matrix, `matrixp`. Recall that, for transition matrices, the four probabilities in each row must sum to 1.0. So if the elements in the i th row of `matrixn` sum to N , then the probability value at each element of the transition matrix, `matrixp`, is $(n_{ai} + 1)/(N + 4)$, $i = 1, 2, 3, 4$, if we apply the small sample correction that we used earlier.

Clearly, all of these operations are much too tedious for hand calculation; therefore, we write a function in R to perform the job for us:

```
transmatp<-function(sites,col,matrix0){
#sites = numeric matrix of n binding sites, w positions
#col = column that transition matrix produces
#matrix0 = matrix of counts for n = col, initialized to 0
matrixn<-matrix0
for(i in 1:length(sites[,1])){
  j<-sites[i,(col-1)]
  matrixn[j,sites[i,col]]<-matrixn[j,sites[i,col]]+1
}
#Change counts to probabilities
matrixp<-matrixn
matrixp<-matrixp+1 #Adds 1 to every element
for(i in 1:4){
  matrixp[i,<-matrixp[i,]/sum(matrixp[i,])
  #Denominator=sum(matrixn[i,])+4
}
return(matrixp, matrixn)
}
```

```
> tmp<-transmatp(gata,2,matrix0)
```

The result of this calculation is shown as an R list object below:

```
> tmp$matrixp
      [,1]      [,2]      [,3]      [,4]
[1,] 0.06060606 0.03030303 0.8484848 0.06060606
[2,] 0.16666667 0.16666667 0.5000000 0.16666667
[3,] 0.20000000 0.20000000 0.4000000 0.20000000
[4,] 0.04761905 0.09523810 0.7142857 0.14285714
```

```
> tmp$matrixn
      [,1] [,2] [,3] [,4]
[1,]    1    0   27    1
[2,]    0    0    2    0
[3,]    0    0    1    0
[4,]    0    1   14    2
```

```
> matrix2<-tmp$matrixp #Identical to $matrixp above
```

`matrix2` is the transition matrix that generates the probability distribution for the second position given the probability distribution for the first position given in `vector1`. For completeness, we present all of the other matrices calculated for the remaining positions. These are computed in the same way as for `matrix2`.

```
> matrix3
      [,1]      [,2]      [,3]      [,4]
[1,] 0.2000000 0.2000000 0.4000000 0.2000000
[2,] 0.4000000 0.2000000 0.2000000 0.2000000
[3,] 0.9375000 0.0208333 0.0208333 0.0208333
[4,] 0.5714286 0.1428571 0.1428571 0.1428571
```

```
> matrix4
      [,1]      [,2]      [,3]      [,4]
[1,] 0.0384615 0.0192308 0.0576923 0.8846154
[2,] 0.2500000 0.2500000 0.2500000 0.2500000
[3,] 0.2000000 0.2000000 0.2000000 0.4000000
[4,] 0.2500000 0.2500000 0.2500000 0.2500000
```

```
> matrix5
      [,1]      [,2]      [,3]      [,4]
[1,] 0.2000000 0.2000000 0.2000000 0.4000000
[2,] 0.2500000 0.2500000 0.2500000 0.2500000
[3,] 0.3333333 0.1666667 0.1666667 0.3333333
[4,] 0.7200000 0.0800000 0.0600000 0.1400000
```

```

> matrix6
      [,1]      [,2]      [,3]      [,4]
[1,] 0.4000000 0.07500000 0.4250000 0.1000000
[2,] 0.4285714 0.14285714 0.1428571 0.2857143
[3,] 0.1666667 0.16666667 0.5000000 0.1666667
[4,] 0.5000000 0.08333333 0.2500000 0.1666667

```

These five matrices, together with `vector1`, provide the Markov chain representation of the GATA-1 transcription factor binding sites listed in Table 9.3. If we wanted, we could use the initial probability distribution `vector1` and these transition matrices to simulate GATA-1 binding sites in a manner similar to the simulation that we did in Section 2.3.3.

One more point needs to be made before we move on: How do we *score* a sequence given a representation of the type shown above? Scoring proceeds in a fashion analogous to the approach used for the PWMs. Convert the probabilities in the initial probability distribution into $\log_2(p_{ai}/q_a)$ using q_a for the genome as a whole (which means that we are comparing sites to an iid model). Convert the probabilities in the transition matrices into $\log_2(p_{ij}/q_{ij})$ values using $q_{ij} = q_j$ (also assuming an iid model).

Let's take a specific example using the variable names and notation employed in the example above. Suppose the sequence to be scored is **TGATAA**. The score s_1 for the first position is $\log_2(p_{a1}/q_a)$, selecting the vector element corresponding to position 1 in the initial probability distribution vector, `vector1`. Since we have coded A, C, G, and T, respectively, as 1, 2, 3, and 4, the element that we require is `vector1[4]` (corresponding to T). The score is $s_1 = \log_2(\text{vector1}[4]/q_T)$. The score for the second position is taken from the transition matrix corresponding to position 2, `matrix2`. The element of `matrix2` whose row corresponds to the base that appears at position 1 (T) and whose column corresponds to the base that appears at position 2 (G) provides the information needed to compute $s_2 = \log_2(\text{matrix2}[4, 3]/q_G)$. Scores at the other positions are calculated similarly.

9.4 Entropy and Information Content

Shannon's entropy is a concept drawn from information theory and signal processing. It measures the degree of uncertainty associated with a set of possible outcomes. Given a discrete random variable X with J outcomes x_1, x_2, \dots, x_J having probabilities $p(x_1), \dots, p(x_J)$, respectively, Shannon's entropy is conventionally defined as

$$H(X) = - \sum_{j=1}^J p(x_j) \log_2 p(x_j), \quad (9.6)$$

where $p(x) \log_2 p(x) = 0$ if $p(x) = 0$.

Now suppose we are looking at a sequence X_1, X_2, \dots of iid bases. The entropy of the i th position is

$$H(X_i) = - \sum_{a \in \{\text{A,C,G,T}\}} p(a) \log_2 p(a). \quad (9.7)$$

If the outcomes are equally probable, $p(a) = 1/4$ and then $H(X_i) = \log_2 4 = 2$. The uncertainty in this case can be represented by two bits. One bit accounts for the possibility that a base is a purine (1) or pyrimidine (0), and the other bit specifies which purine (1 or 0, given that the first bit was 1) or which pyrimidine (1 or 0, given that the first bit was 0) is present. If we know that the base at a position is actually an A, for example, then $p(a) = 1$ if $a = \text{A}$, and all other $p(a) = 0$. In this case, $H(X_i) = 0$ (i.e., the uncertainty is zero).

The **information** is a measure of how much the entropy is reduced after a “signal” (in this case, resulting from natural selection at a position within a region of DNA) has been received:

$$I(X_i) = H_{\text{before}} - H_{\text{after}}. \quad (9.8)$$

If the bases at a position had initially been distributed with equal probabilities, and over the course of time had evolutionarily been fixed to be an A, then using the calculation above, we see that the information at that position is now $I(X_i) = 2$ bits.

Another measure of information content (which becomes identical to Shannon’s entropy when the probability distribution is uniform) is the **relative entropy**, or Kullback-Leibler distance. We start with a set of aligned sites, where, as before, the probability of finding base a at position i is given by p_{ai} . Let q_a represent the background distribution of bases in a genome or in a random model of a genome. The relative entropy is then defined as

$$H(\mathbf{p}_i || \mathbf{q}) = \sum_{a \in \{\text{A,C,G,T}\}} p_{ai} \log_2(p_{ai}/q_a). \quad (9.9)$$

The $||$ indicates that a *distribution* \mathbf{p}_i is being examined relative to a distribution \mathbf{q} . Using the log-odds scoring scheme that we employed above, we can calculate the expectation for the score at any position in the sequence:

$$\mathbb{E}(S_i) = \sum_a p_{ai} s_a = \sum_a p_{ai} \log_2(p_{ai}/q_a). \quad (9.10)$$

In this expression, the p_{ai} correspond to the distribution at position i . We see that the expected value for the score at any position is the same as the relative entropy at that position. This provides an intuitive view of the nature of relative entropy.

Why have we introduced relative entropy and information content? The major reason is to help define the *extent* of signals. Remember that an experimental measure of the extent of a binding site comes from footprinting

experiments. But how can we decide the extent of a signal in probabilistic terms given a set of sequences containing that signal? One approach is to plot information content as a function of position along the set of sequences. The binding site boundary is taken as that position where the relative entropy exceeds a particular threshold (e.g., the value of relative entropy exceeded by no more than 5% of the positions in iid sequences).

The informational perspective has led to a depiction of signals as sequence logos (Schneider and Stephens, 1990). A **sequence logo** is a graphical representation of a signal in which the total height at each position corresponds to the relative entropy. The height of each *letter* at each position is calculated by multiplying the relative entropy at that position by the frequency of the corresponding letter. A logo representing the lambda operator sites (Table 9.1) is shown in Fig. 9.6. Logos are better visual representations than consensus sequences because the logos indicate both the amount of relative entropy at each position (the height of the stack of four letters) and the relative contribution of each base (the relative height of the letter) at that position.

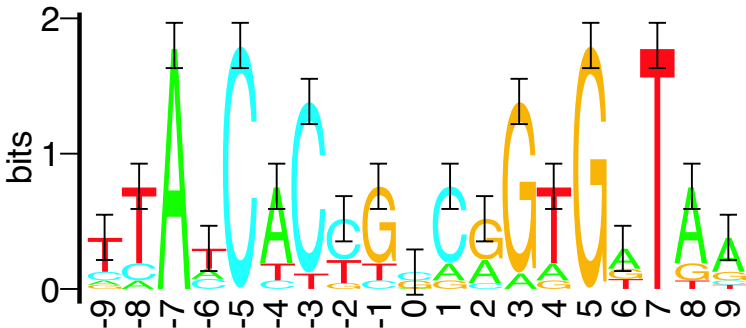


Fig. 9.6. [This figure also appears in the color insert.] DNA sequence logo of lambda operator sites. The illustration includes one more position to the right and left of the sequences shown in Table 9.1. Heights of each letter A, C, G, or T (in bits) represent the contribution of each letter to the information encoded at each position. The sequence logo was provided by Dr. Thomas D. Schneider, National Cancer Institute (<http://www.lecb.ncifcrf.gov/~toms/papers/hawaii/lambdaacro/>).

9.5 Signals in Eukaryotic Genes

Signals are important components of eukaryotic genes. Examples of promoter signals, transcriptional signals, and mRNA processing signals are provided in Table 9.4. In Section 9.2, we represented the consensus sequences by writing

down the actual base that is overwhelmingly preferred at each position. In Table 9.4, we expand the representation by using single-letter codes to describe different *combinations* of bases. For example, (A/T) is represented by W, and (A/G) is represented by R. (See Appendix C.1 for a complete list of the IUPAC-IUB codes.) With this notation, the GATA-1 consensus can be more flexibly represented as **WGATAR**. Lowercase letters are sometimes used to depict a weak preference for a particular base, with all others appearing at lower frequencies. The last two positions in the 3' splice signal are designated by this notation. These types of designations are not appropriate for gene finding because we need to be able to generate a score. Positional weight matrices (also called **profiles** in this context) or Markov chains are effective representations of these signals. Examples of these are presented in Chapter 14 (Table 14.3).

Table 9.4. Examples of signals and their consensus sequences in human genes. For further details about the TATA box see Milanesi and Rogozin (1998) and for the others see Zhang (1998).

Site	Consensus ¹																
TATA box	-3	-2	-1	0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+10	+11		
	S	T	W	T	W	W	A	W	R	S	S	S	S	S	S		
CAAT box	-4	-3	-2	-1	0	+1	+2	+3	+4								
	R	R	C	C	A	R	K	S	R								
Cap site	-2	-1	0	+1	+2	+3	+4	+5									
	K	C	W	S	Y	S	S	S									
Start site	-9	-8	-7	-6	-5	-4	-3	-2	-1	0	+1	+2	+3				
	S	S	S	S	S	C	R	M	C	A	T	G	R				
5' splice signal	-3	-2	-1	0	+1	+2	+3	+4	+5								
	N	A	G	G	T	R	A	G	W								
3' splice signal	-15	-14	-13	-12	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1	0	+1
	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	N	Y	A	G	g t

¹ Weaker preferences are indicated by lowercase letters. Locations with preferences for particular combinations of bases are indicated by IUPAC-IUB codes. In addition to A, C, G, T, they are R = A or G (purine), Y = T or C (pyrimidine), S = C or G, W = A or T, K = G or T, M = A or C, and N = any base.

Regulatory sequences control the timing and levels of gene expression by binding specialized transcription factors. The regulatory regions of genes can extend tens of thousands of base pairs upstream of the transcriptional start site, and some sequences (enhancers) can be found either upstream or downstream of the promoter (sometimes within introns or downstream of the polyadenylation site). Probabilistic descriptions can be used as an aid to locating these binding sites (e.g., the GATA-1 binding site; Section 9.3). Numbers and locations for k -tuples corresponding to signals such as GATA-1 may provide similar information about signal content. However, many binding sites

are quite short, and therefore they are likely to occur by chance in long lengths of sequence.

Our probabilistic description has not taken into account the *distributions* of the sites along the DNA although typical values for locations of GC boxes, CAAT boxes, and TATA boxes relative to the transcriptional start sites are known (Chapter 14.4.2). The spacing (or alternatively density) of candidate transcription factor binding sites is additional information that can be used for assessing whether they have been correctly identified. This sort of information can be incorporated in *hidden Markov models* (not discussed in this book). Clearly, signals are only one set of features that describe eukaryotic genes, but they are an important component of gene-finding tools. Eukaryotic gene finding is discussed at greater length in Chapter 14.

9.6 Using Scores for Classification

We have described how to represent signals in probabilistic terms and how to produce scores for instances of any signal. The scores can be used to classify any candidate string into one of two categories: sites or nonsites. There are two types of errors that can result from this procedure. Let the **null hypothesis** \mathcal{H} be that the sequence to be tested is a site. A **Type I error** is one that classifies a site as a nonsite (i.e., a false negative, rejecting \mathcal{H} when it is true). A **Type II error** is one that classifies a nonsite as a site (a false positive, failing to reject \mathcal{H} when it is false):

\mathcal{H} is:	Assigned class is:						
	<table style="width: 100%; border-collapse: collapse;"> <tr> <td style="border-bottom: 1px solid black; padding: 2px 10px;">True</td> <td style="border-bottom: 1px solid black; padding: 2px 10px;">False</td> </tr> <tr> <td style="padding: 2px 10px;">True</td> <td style="padding: 2px 10px;">Type I error</td> </tr> <tr> <td style="padding: 2px 10px;">False</td> <td style="padding: 2px 10px;">Type II error</td> </tr> </table>	True	False	True	Type I error	False	Type II error
True	False						
True	Type I error						
False	Type II error						

The performance of a classification method is often described in terms of **sensitivity** (S_n , the proportion of actual features detected) and **specificity** (S_p , the proportion of predicted features that are real). In terms of Type I and Type II errors, we have

$$S_n = 1 - \mathbb{P}(\text{Type I error}),$$

$$S_p = 1 - \mathbb{P}(\text{Type II error}).$$

If #TP is the number of true positive predictions, #FP is the number of false positive predictions, #TN is the number of true negative predictions, and #FN is the number of false negative predictions, then

$$S_n \approx \frac{\#TP}{\#TP + \#FN}, \quad S_p \approx \frac{\#TN}{\#TN + \#FP}. \quad (9.11)$$

Given a training set of sites, scoring produces a distribution of scores for these sites, shown as an idealized distribution, A, in Fig. 9.7. Given a set of sequences that are not sites, the same scoring method produces another distribution, B, that in general overlaps the first one. We can place a *cutoff score* C on the graph and classify any sequence with score $S \geq C$ as a site and classify any sequence having $S < C$ as a nonsite. The area of distribution A to the left of C represents the fraction of false negative assignments, and the area of distribution B to the right of C is the fraction of false positive assignments. We may move the cutoff lower in an effort to avoid “losing” actual sites, but we do so at the expense of a greater number of false-positive assignments. We can move the cutoff to the right to reduce the false-positive assignments, but then we lose sensitivity.

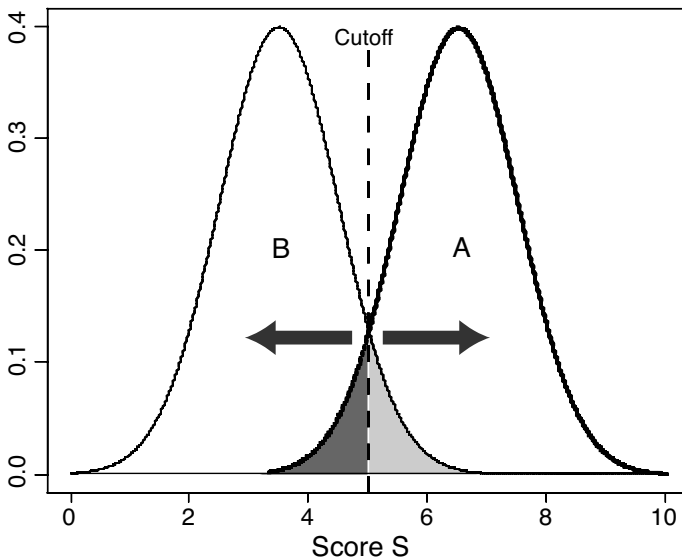


Fig. 9.7. Idealized illustration of the classification of objects of types A and B based upon their score distributions. An arbitrary cutoff score midway between the maximum scores for the two object types is indicated. Objects B having scores larger than the cutoff score (light shaded area) are incorrectly classified as A, while objects A having scores below the cutoff score (dark shaded area) are incorrectly classified as B. If objects are classified to answer the question of whether they are objects of type A, then the light shaded area corresponds to false positive assignments and the dark shaded area corresponds to false negative assignments. The cutoff score can be adjusted up or down (arrows) to minimize false positive assignments (with correspondingly greater false negative errors) or to minimize false negative assignments (with correspondingly greater false positive errors).

In actual practice, we may be scanning an entire genome for a limited number of sites (e.g., one site per 10^3 to 10^4 bp). In this case, the number of nonsites examined is equal to $(G - w) - n$, where G is the number of base pairs in the genome, w is the length of the site, and n is the number of actual sites. Ordinarily, $G \gg n$, so many more nonsites than sites are scored. The result is that if we set the cutoff midway between the scores corresponding to the maxima of the two curves, the *number* of false positive assignments will vastly exceed the *number* of actual sites. For this sort of application, we might accept a lower sensitivity to achieve higher specificity. If, however, we are only analyzing the upstream region of a particular gene of interest, then we might move the cutoff to the left to increase the sensitivity at the expense of specificity. We make the trade-offs necessary for the problem at hand. We illustrate these principles in Computational Example 9.3.

Another approach to assessing a classification scheme uses the **false discovery rate (FDR)**, defined to be the expected proportion of false positive features among those features called positive. This is given by

$$\text{FDR} \approx \frac{\#\text{FP}}{\#\text{FP} + \#\text{TP}}.$$

For details on the calculation of the FDR, see, for example, Storey and Tibshirani (2003).

Computational Example 9.3: Classification of sites using GATA-1 PWM

We use the PWM computed in Computational Example 9.1 to illustrate the classification problem. Remember that we had a training set comprising 49 sites and that these represent a sample of all possible sites. First, we employ our PWM to simulate 5000 sites and score them to determine the site score distribution implied by our PWM. Next, we simulate a set of 5000 “background” sequences of the same length as GATA-1 sites (6 bp) by using the iid model, and we determine the score distribution of this set. Finally, we examine FP and FN error rates at different cutoff scores.

Step 1: Simulating GATA-1 sites

Our R function `makepwm` produced a matrix of probabilities at each position as one of its two returned objects. This is just what we need to simulate the sites. The function to do this is:

```
simmotif<-function(pwm){
# pwm is a PWM matrix of probabilities (4xL)
L<-length(pwm[1,]) #Number of positions in the motif
motif<-rep(0,L) #Create and initialize motif vector
dna<-c(1,2,3,4) #Numeric codes for A, C, G, T
```

```

for (j in 1:L){
  motif[j]<-sample(dna,1,p=pwm[,j])
}
return(motif)
}

```

We have used `sample` before, so this function should be clear to you. The only wrinkle is that for each cycle of the `for` loop, a different probability distribution is used, corresponding to each column of `pwm`. The need for these probability distributions was anticipated when we devised the `makepwm` function. To simulate N sites, we make an $N \times 6$ matrix and then run `simmotif` N times. Remember that the `pwm` argument that we need is in `tmp` as `tmp$pwm`.

```

> N<-5000
> gata.motifs<-matrix(nrow=N,ncol=6)
> for(i in 1:N){
+   gata.motifs[i,]<-simmotif(tmp$pwm)
+ }

```

We check some of the simulated sites to make sure that they look reasonable (compared with the `gata` object).

```

> gata.motifs[1:5,]
  [,1] [,2] [,3] [,4] [,5] [,6]
[1,]   4   3   1   4   1   3
[2,]   1   3   1   4   1   3
[3,]   4   3   1   4   1   3
[4,]   1   3   1   4   4   3
[5,]   1   4   1   4   3   3

```

Step 2: Scoring simulated sites

We now wish to generate the score distribution for the simulated sites. What would we predict for this distribution, based on the training set scores (Fig. 9.3)? We can employ the `calcscore` function operating on each row of `gata.motifs` to obtain the scores:

```

> gata.motifs.score<-rep(0,N)
#vector to hold the results of computation
> for(i in 1:N){
+   gata.motifs.score[i]<-
+   calcscore(gata.motifs[i,],log2pwm)
+ }

```

We can compare this set of scores to the training set by looking at the means:

```

> mean(gata.motifs.score)
[1] 5.094539
> mean(gata.score)
[1] 6.07064

```

At first, it may seem strange that the simulated motif scores have a smaller average than those of the training set, but things become clearer when we examine the score distributions. We produce a histogram of `gata.motifs.score` in the usual manner, with the result shown in Fig. 9.8 (solid lines: code supplied later).

Notice that our simulated sites have a score distribution that is skewed left (as is the training set distribution), and note that the central part of the distribution has been filled in. This accounts for the lower mean value for the scores. Remember that the training set is a sample of the set of actual sites. As an exercise, you can test how much individual samples vary by repeatedly sampling 49 of the simulated set of 5000 motifs, recomputing the corresponding PWMs, and plotting the resulting scores for each sampling (see Exercise 13 at the end of this chapter) .

Step 3: Simulating the background

We now simulate sequence strings of length N according to the iid background model. The function to do this is:

```

simbg<-function(bg,L){
  #bg is a vector of probabilities for A, C, G, T (1x4)
  #L = length of sites to be simulated
  seq<-rep(0,L)
  dna<-c(1,2,3,4) #Numeric codes for DNA
  seq<-sample(dna,L,replace=T,p=bg)
  return(seq)
}

```

We apply this function 5000 times in the same manner as we did with `simmotif`, and examine the resulting score distribution.

```

> for(i in 1:N){
+   back.sim[i,]<-simbg(bg,6)
+ }
> back.sim.score<-rep(0,N)
> for(i in 1:N){
+   back.sim.score[i]<-calcscore(back.sim[i,],log2pwm)
+ }

```

Now we examine the score distribution for the background and compare it with the distribution for the simulated motifs:

```
> hist(back.sim.score, prob=T,xlim=c(20,20),
+ ylim=c(0,0.25),lty=2)
> hist(gata.motifs.score, xlim=c(-20,20),
+ prob=T,lty=1,add=T)
```

The distribution of the background scores (Fig. 9.8, broken lines) is more nearly symmetric, and the mean is at -6.56 . In Computational Example 9.2, we asked what we might have expected the mean score for the background distribution to be. In particular, why is it not 0? The answer comes from looking at `tmp$log2pwm`: 16 of the 24 matrix elements are negative, and their average value is -2.32 . Eight of the 24 matrix elements are positive, and their average value is 1.14 . The iid sites draw more of their positional scores s_i from the negative values.

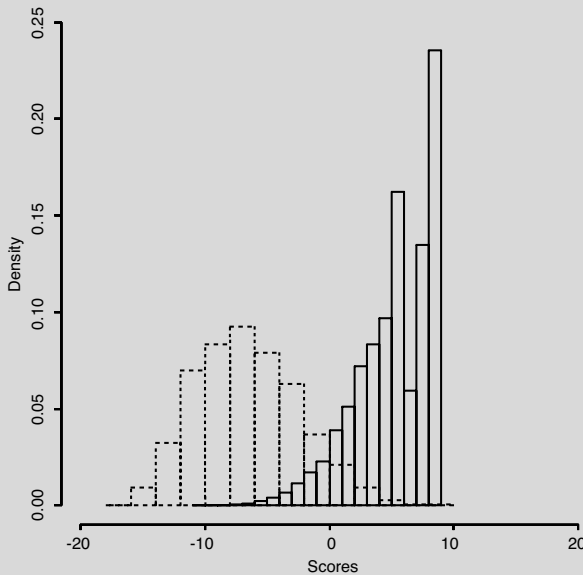


Fig. 9.8. Score distributions for GATA-1 sites (solid lines) simulated from the PWM corresponding to sites in Table 9.3 (scores shown in Fig. 9.3) and for “background” sequences (broken lines) simulated from an iid model using the base composition of human DNA.

Now we can examine the false positive and false negative error rates. To do this, we specify a set of cutoffs from -10 to $+8$ (encompassing most of the scores in `gata.motifs.score`) and calculate the fraction of `gata.motifs.score` values *below* the cutoff (false negative) and the fraction of `back.sim.score` values *above* the cutoff (false positive).

```
cutoffs<-c(-10:8)
false.neg<-rep(0,19) #Vector to hold calculated values
```

```

for(i in 1:19){
  false.neg[i]<-length(gata.motifs.score[gata.motifs.score[]
    <cutoffs[i]])/N}

false.pos<-rep(0,19) #Vector to hold calculated values
for(i in 1:19){
  false.pos[i]<-length(back.sim.score[back.sim.score[]
    >cutoffs[i]])/N}

> plot(cutoffs,false.neg,type="l",xlim=c(-10,10), ylim=c(0,1))
> points(cutoffs,false.pos,type="l",lty=2)

```

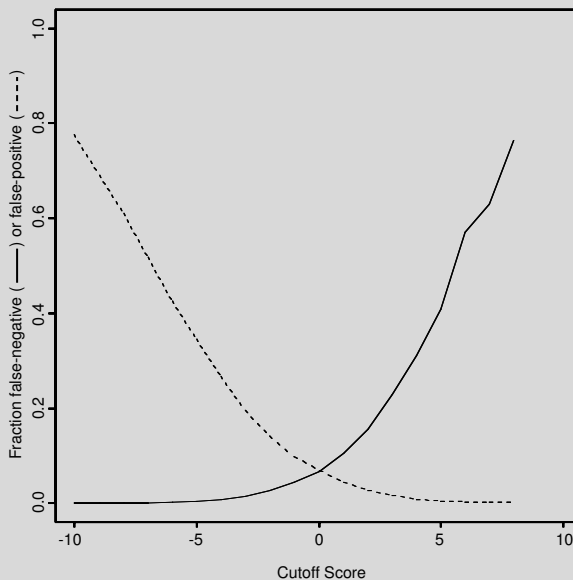


Fig. 9.9. False negative errors (fraction of simulated GATA-1 sites identified as “background”: solid line) and false positive errors (fraction of simulated “background” sequences identified as GATA-1 sites: broken line) for simulated motifs and background sequences as a function of cutoff score. The histogram of scores is shown in Fig. 9.8.

The plot in Fig. 9.9 shows the expected opposite behavior of the false positive and false negative error rates. If we were to choose a cutoff score of 0.0, we would have a false negative rate of 6.6% and a false positive error rate of 6.8%. This latter value is deadly for scanning long sequences. For example, if we were to examine 100,000 base pairs upstream of a particular gene, we would incorrectly predict approximately 6800 GATA-1 sites (assuming that human DNA is adequately represented by the iid model).

As mentioned above, to avoid the “noise” associated with the false positive error, we might choose to increase the cutoff and accept the concomitant false negative errors. Ultimately, we might choose a more sophisticated probabilistic model that includes the expected site distribution (e.g., a hidden Markov model), but this is beyond the scope of this chapter.

References

- Brazma A, Jonassen I, Vilo J, Ukkonen E (1998) Predicting gene regulatory elements in silico on a genomic scale. *Genome Research* 8:1202–1215.
- Bussemaker HJ, Li H, Siggia ED (2000) Building a dictionary for genomes: Identification of presumptive regulatory sites by statistical analysis. *Proceedings of the National Academy of Sciences USA* 97:10096–10100.
- Durbin R, Eddy S, Krogh A, Mitchison G (1998) *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids*. Cambridge: Cambridge University Press.
- Gussin GN, Johnson AD, Pabo CO, Sauer RT (1983) Repressor and Cro protein: Structure function, and role in lysogenization . In Hendrix RW, Roberts JW, Stahl FW, Weisberg RA (eds.), *Lambda II*. Cold Spring Harbor, NY: Cold Spring Harbor Laboratory, pp. 93–121.
- Harley CB, Reynolds RP (1987) Analysis of *E. coli* promoter sequences. *Nucleic Acids Research* 15:2343–2361.
- Kissinger CR, Liu BS, Martin-Blanco E, Kornberg TB, Pabo CO (1990) Crystal structure of an engrailed homeodomain-DNA complex at 2.8Å resolution: A framework for understanding homeodomain-DNA interactions. *Cell* 63:579–590.
- Milanesi L, Rogozin IB (1998) Prediction of human gene structure. In Bishop MJ (ed.) *Guide to Human Genome Computing* (2nd edition) San Diego: Academic Press, pp. 213–259.
- Schneider T. A gallery of sequence logos.
<http://www.lecb.ncifcrf.gov/toms/sequencelogo.html>
- Schneider TD , Stephens RM (1990) Sequence logos: A new way to display consensus sequences. *Nucleic Acids Research* 18:6097–6100.
- Storey JD, Tibshirani R (2003) Statistical significance for genomewide studies. *Proceedings of the National Academy of Sciences USA* 100:9440–9445.
- Stormo GD (1990) Consensus patterns in DNA. *Methods in Enzymology* 183:211–221.
- Stormo GD (2000a) DNA binding sites: representation and discovery. *Bioinformatics* 16:16–23.
- van Helden J, Andre B, Collado-Vides J (1998) Extracting regulatory sites from the upstream region of yeast genes by computational analysis of oligonucleotide frequencies. *Journal of Molecular Biology* 281:827–842.

Zhang MQ (1998) Statistical features of human exons and their flanking regions. *Human Molecular Genetics* 7:919–932.

Exercises

Exercise 1. Write out a duplex DNA sequence that contains the sequence pattern 5'-AGAACA-3' as an inverted repeat. Separate the centers of inverted repeats by a number of base pairs that places corresponding positions in the pattern on the same side of the DNA helix.

Exercise 2. Generalize (9.5) to the case for which not all q_{ai} are equal. Will a single equation suffice?

Exercise 3. Make a plot of S_n and S_p (9.11) versus cutoff scores for simulated GATA-1 sites using an iid background (see Computational Example 9.3 and Fig. 9.8). Add the FDR curve to your plot.

Exercise 4. If the samples of sites or non-sites are sufficiently large that small sample effects are negligible, do S_n and S_p depend upon sample size? Does the false discovery rate depend upon sample size (i.e., does the computed FDR apply to a *particular* analysis or is it a general property for a given cutoff)?

Exercise 5. From the data in Table 9.1, produce a PWM describing the lambda operator half-sites, and generate a score for each site. Make a histogram of the score distribution.

Exercise 6. Use the matrix of probabilities describing the *E. coli* -10 sequence TATAAT (Section 9.2.1 and the R function in Computational Example 9.3) to simulate ten such sequences.

Exercise 7. Plot the relative entropy as a function of position for the human start site represented by the profile or PWM in Table 14.3. Then produce (by hand, to make sure you understand the principles) a sequence logo representing this site.

[Hint: One way of creating letters with adjustable widths and heights is to take screen shots of enlarged letters A, C, G, and T. These can be pasted into a Microsoft Word document and scaled as needed.]

Exercise 8. Use your PWM from Exercise 5 to simulate 100 lambda operator half-sites, and also simulate 100 non-sites of the same length using iid letters with 50% G+C. Create a PSSM for the simulated half-sites. Score the simulated half-sites and the simulated non-sites, and plot the histograms for each. Select a cutoff score for discriminating between sites and non-sites, and justify your choice.

Exercise 9. Generate 48,502 bp of iid sequence, 50% G+C (see Chapter 2). Using the PWM from Exercise 8, generate scores for every starting position of a potential lambda operator half-site using an R function that you create. Make a histogram of scores obtained from this sequence, and estimate the number of half-sites that you would have predicted. [Hint: Compare the histogram from this problem with the one that you obtained in Exercise 8.]

Exercise 10. Repeat Exercise 9, except instead of using a simulated sequence, use the lambda DNA sequence in GenBank file NC_001416. This can be downloaded in FASTA format from the NCBI Web site (Appendix B).

Exercise 11. Calculate by hand the transition matrix needed to produce the probability distribution at position 3 for the GATA-1 sites listed in Table 9.3.

Exercise 12. Write an R function for producing the score for an arbitrary sequence using the Markov chain representation of GATA-1 sites in Computational Example 9.2.

Exercise 13. To illustrate the effects of sampling variation, take three independent random samples each containing 49 simulated GATA-1 sites from the 5000 sites produced in Computational Example 9.3. Compute the three corresponding PWMs, and use each PWM to generate scores for the actual sites in Table 9.3. Plot the resulting score distributions and calculate the mean score corresponding to each PWM.

Exercise 14. The code at the end of this exercise will produce all sequence strings having two differences from a specified input string.

- a. Write an R function that will produce all sequences differing from a given sequence at exactly one position.
- b. Generate all sequences that are identical to the promoter -35 consensus TTGACA or that differ from it at one or at two positions. Use all of these to create a PWM for the -35 sequence.
- c. Use the PWM from part b of this exercise to identify the -35 sequences in promoters for *caiT*, *damP2*, *dnaQP2*, *gapB*, *melA*, and *nrd*.

Code:

```
neighbor2<-function(x){
#function to make all 2-neighbors
#x is the input string
out.file<-x #holds result
#First change: 1-neighborhood specification
for(j in 1:6){
  for(k in 1:4){
    y<-x
```

```

        if(k!=x[j]){
            y[j]<-k
#Second change: becomes 2-neighborhood specification
        for(m in 1:6){
            if(m!=j){
                for(n in 1:4){
                    z<-y
                    if(n!=x[m]){
                        z[m]<-n
                                out.file<-rbind(out.file,z)
                }}}
        }}}
    out.file<-out.file[2:length(out.file[,1]),]
    return(out.file)
#contains sequence variants with two changes
}

```

Exercise 15. A combinatorial approach to the previous problem is possible. Show that there are 16 five-letter words in the one-mismatch neighborhood of GCATC (including GCATC itself). How many of these words begin with G? How many begin with a letter other than G? Use this information to produce the first column of a PWM.

Exercise 16. Indicate how PWMs may be used to define neighborhood sequences (Section 7.4.1) to be used for identifying approximate “hits” in the rapid search method described in Section 7.4.1. Produce the PWM corresponding to the word GCATC with a neighborhood size of one mismatch (Exercise 15), and state the threshold score T above which a hit (within one mismatch) to GCATC will be declared.

Similarity, Distance, and Clustering

10.1 The Biological Problem

In this chapter, we explore quantitative approaches to **clustering**, the process of *identifying groups* of like objects. This grouping is based upon similarities or differences as measured by the characters that the objects possess. Clustering is closely related to the process of **classification**, which is *assigning objects* into predetermined categories. This assignment to a category is also based upon the particular states of the characters associated with that **object**. We discussed classification in the last chapter and will say a little more at the end of this chapter. For more extensive discussions of clustering and classification, see Dunn and Everitt (1982), Everitt and Dunn (2001), and Johnson and Wichern (2002).

Clustering and classification have a long history in the biological sciences. About 1.7 million biological species have been described, and tens of millions of species are thought to exist. Nevertheless, biological diversity becomes comprehensible because organisms can be classified *hierarchically* into groups. This exercise was actively pursued during the eighteenth century by Carolus Linnaeus, who is well-known for his system of biological nomenclature. By hierarchical classification, we mean that groups of similar organisms (such a group being called a **taxon**—plural: **taxa**) are subsets of larger groupings. For example, mammals are a subset of tetrapods (amphibians, reptiles, mammals, and other organisms having four appendages), tetrapods are a subset of vertebrates (animals with backbones), and vertebrates are a subset of deuterostomes (animals whose mouth is formed second during the gastrulation stage of embryonic development). Classification of organisms is still practiced today by biologists involved in systematics and by paleontologists. For example, in classifying dromaeosaurs (bipedal dinosaurs such as *Velociraptor*), more than a hundred skeletal characters (e.g., relative sizes of premaxillary teeth, fusion of tarsal bones) may be employed.

Clustering and classification are equally important for genomic analysis. For example, spotted microarray or oligonucleotide array technologies are used

to analyze gene expression as measured by mRNA abundances in cells. In this case, the objects being clustered usually are genes, and the characters are relative mRNA abundances measured under particular conditions. In particular, animal cells in tissue culture might be deprived of serum and then, after serum is added back, the levels of mRNA species (relative to levels in cells not deprived of serum) are measured at different time points. Or, the objects might be genes in human cells, and the characters might be relative mRNA expression levels in normal cells, cancer cells, or both cell types treated with an antitumor agent. The purpose of clustering in this case is to identify and group together (cluster) genes having similar expression patterns. Similar expression patterns may indicate that the genes participate in similar biological processes or that they respond to similar biological controls.

Data in the biological world are commonly multivariate. This means that biological objects and phenomena are associated with a number of variables, each of which contributes to the observed phenomenon. In both cases described above, the data consist of a matrix of m rows of objects (conventionally called **operational taxonomic units**, or **OTUs**, in evolutionary studies) and p columns of characters describing those objects. The distinctions between different **objects** or operational taxonomic units are based upon the states of their differing characters. These **characters** are properties that can take on different values and can differ among OTUs (for example, fused metatarsals or relative mRNA abundance 30 minutes after addition of serum). There may be on the order of 10^4 objects (genes), each described by states of 10–100 characters. The list of states for each character that describes an object might be thought of as a vector, and thus clustering and classification may require descriptions using high-dimensional vector spaces. Such complex multivariate data require appropriate quantitative approaches.

As an aside, note that clustering and classification are important in fields other than biology. For example, businesses as represented by their stocks may be classified into different “taxa,” such as “small cap. value” or “large cap. growth,” based upon their asset base, business model, and other factors. For financial lending, criteria (characters) pertaining to prospective borrowers (e.g., employment, income level, credit history) are employed to generate a “credit score,” which determines the class into which the prospective borrower will be classified (credit-worthy or not credit-worthy).

10.2 Characters

Before we discuss measures of similarity or difference and clustering, we need to examine different types of characters.

1. **Qualitative or categorical characters** differ in *type*. For example, the coat color of mice might be either black (2), brown (1), or white (0). As a further example, at any particular position in a DNA sequence,

the state at that position could be **A**, **C**, **G**, or **T** (coded as 1, 2, 3, or 4, respectively). Even though these states may be coded numerically, no *arithmetic* operations (such as multiplication or division) can be applied to them.

2. **Quantitative characters** are measured on a numerical scale, and these may take on either **discrete** values or **continuous** values. For example, the number of hydrogen bond donor or acceptor positions in the major groove at any position in the DNA will be an integer (discrete). Among fossil dinosaurs, the number of caudal (tail) vertebrae in a dromaeosaur skeleton will be an integer (discrete), whereas the length of the tail in centimeters will be drawn from a continuous distribution.
3. **Dichotomous characters** may take one of two possible values or states. For example, if the character is the presence or absence of a Y chromosome in human cells (qualitative or categorical), then only two possible values are possible (counting multiple copies of Y in XYY individuals as the “presence of Y”). Similarly, the character state at any position in a DNA sequence is either a purine or a pyrimidine. It is possible to convert other types of quantitative characters into dichotomous ones. For example, if we were studying arthropods (insects, spiders, crustaceans, millipedes), we could convert leg numbers into two character states: state 0 (number of legs not equal to 6) and state 1 (number of legs equal to 6). These states could be used to distinguish insects from the other arthropod classes.

When dealing with biological strings I and J , the character states will correspond to the identities of the letters at each position in each string, and I and J will be said to be similar, with similarity coefficient s_{IJ} determined by the number of character states that match. We may consider strings representing nucleic acid or protein sequences. An example is shown in Table 10.1. This shows alignments of portions of primate cytochrome oxidase subunit II DNA sequences for different pairs of primates. A * indicates positions where the state in the lower member of the pair is identical to the corresponding state in the string at the top. Positions where the bottom string differs from the top one are indicated by listing the different states at the appropriate positions. In this figure, the number of differences and the fractional number of differences D are listed below each pairwise comparison. The **edit distance**, or **Levenshtein distance**, is the minimum number of indels or substitutions required to transform one string into another. The number of differences listed below each pair corresponds to the edit distance. Obviously, these strings (portions of corresponding genes) are similar: How should we describe the similarity? This depends upon the purpose, which is often driven by the biological application.

The amino acid sequences corresponding to the gene regions listed in Table 10.1 are shown in Table 10.2. Note how the amino acid representation differs from the representation of the sequences as DNA. First, the $C \rightarrow T$ transition distinguishing Hy from Pa, Go, and Ho at position 6 (Table 10.1)

Table 10.1. Comparisons between a selected region of cytochrome oxidase subunit II coding sequences (bp 121–180) for different pairs of primates. Hy: *Hylobates* (gibbon); Pa: *Pan* (chimpanzee); Go: *Gorilla*; Ho: *Homo*; Po: *Pongo* (orangutan).

Hy	GCCCTTTCCTAACACTCACAACAAAACCTAACCAACACTAACATTACGGATGCCCAAGAA
Pa	GCCCTTTTCCTAACACTCACAACAAAACCTAACTAATACTAGTATTTTCAGACGCCCAGGAA
Go	GCCCTTTTCCTAACACTCACAACAAAGCTAACTAGCACCAACATCTCAGACGCCCAAGAA
Ho	GCCCTTTTCCTAACACTCACAACAAAACCTAACTAATACTAACATCTCAGACGCTCAGGAA
Po	GCCCTTTTCCTAACACTCACAACGAAACTCACCAACACTAACATCTCAGATGCCCAAGAA
Hy	GCCCTTTCCTAACACTCACAACAAAACCTAACCAACACTAACATTACGGATGCCCAAGAA
Pa	*****T*****T****GT***T*A**C*****G***
	(9 differences; fractional difference = 0.150)
Hy	GCCCTTTCCTAACACTCACAACAAAACCTAACCAACACTAACATTACGGATGCCCAAGAA
Go	*****T*****G*****T*G***C****CT*A**C*****
	(9 differences; fractional difference = 0.150)
Hy	GCCCTTTCCTAACACTCACAACAAAACCTAACCAACACTAACATTACGGATGCCCAAGAA
Ho	*****T*****T**T*****CT*A**C**T**G***
	(9 differences; fractional difference = 0.150)
Pa	GCCCTTTTCCTAACACTCACAACAAAACCTAACTAATACTAGTATTTTCAGACGCCCAGGAA
Go	*****G*****GC**C*AC**C*****A***
	(8 differences; fractional difference = 0.133)
Pa	GCCCTTTTCCTAACACTCACAACAAAACCTAACTAATACTAGTATTTTCAGACGCCCAGGAA
Ho	*****AC**C*****T*****
	(4 differences; fractional difference = 0.067)
Go	GCCCTTTTCCTAACACTCACAACAAAGCTAACTAGCACCAACATCTCAGACGCCCAAGAA
Ho	*****A*****AT**T*****T**G***
	(6 differences; fractional difference = 0.100)

makes no difference in the amino acid sequence because of degeneracy of the genetic code (see Chapter 1). Second, the percentage differences between the different pairs is *not* the same at the amino acid sequence level as at the DNA sequence level. In particular, the fractional difference between Hy and Ho based on these 20 amino acid residues is 0.05, while the difference corresponding to 60 nucleotide residues is 0.15. This is because the gene is under selection (i.e., selection tends to conserve the sequence of amino acids), and synonymous changes—those codon sequence changes that do not change the specified amino acid—are not visible in the amino acid sequence.

When dealing with protein sequences, we might elect to count not amino acid differences but rather the *minimum* number of base changes needed to

Table 10.2. Amino acid sequences from portions of cytochrome oxidase subunit II from selected primates indicated in Table 10.1.

```

Hy  ALFLTLTKLTNTNITDAQE
Pa  ALFLTLTKLTNTSISDAQE
Go  ALFLTLTKLTSTNISDAQE
Ho  ALFLTLTKLTNTNITDAQE
Po  ALFLTLTKLTNTSISDAQE

```

Pairwise alignments:

```

Hy  ALFLTLTKLTNTNITDAQE      Pa  ALFLTLTKLTNTSISDAQE
Pa  *****S*S*****      Go  *****S*N*****
(2 differences = 0.10)      (2 differences = 0.10)

Hy  ALFLTLTKLTNTNITDAQE      Pa  ALFLTLTKLTNTSISDAQE
Go  *****S*S*S*****      Ho  *****N*****
(2 differences = 0.10)      (1 differences = 0.05)

Hy  ALFLTLTKLTNTNITDAQE      Go  ALFLTLTKLTSTNISDAQE
Ho  *****S*****          Ho  *****N*****
(1 differences = 0.05)      (1 differences = 0.05)

```

convert one residue to another. This might correspond to a measure of evolutionary distance between two sequences. But this might not reveal the true differences. Conversion of Asn to Ser (residue 14 in the Hy/Pa comparison in Table 10.2) could have been accomplished by one change: AAC \rightarrow AGC. In actuality, there were two changes: AAC \rightarrow AGT (Table 10.1).

Finally, note that what may be relevant biochemically are the *chemical properties* of the residues. There are only two types of differences in the amino acid sequence shown in Table 10.2: Asn replaces Ser (or vice versa), and Ser replaces Thr. Asn, Ser, and Thr are all uncharged polar amino acids. These changes are expected to make little difference in the physicochemical properties of the respective proteins. Despite the differences at the nucleic acid sequence level and at the amino acid sequence level, each of these protein regions has an identical string of physicochemical properties,

$$\text{nnnnpnpp+nppppnp-np-}$$

where n = nonpolar, p = polar, + = positively charged, and - = negatively charged amino acid residues.

The obvious conclusion is that there are choices to be made for characters used to describe objects, and these choices will be dictated by the purpose for which clustering and classification are undertaken. For example, protein homologs that diverged from a common ancestor at a very remote time may be hard to recognize based upon DNA sequence, easier to recognize based

upon protein sequence, and easier still to recognize based on protein structure (determined by physical properties of constituent amino acids).

10.3 Similarity and Distance

Similarity and distance are measures of how closely- or distantly-related objects are based upon a collection of characters that describe those objects. To illustrate these concepts, we consider an example with which we are all familiar: relationships among some common animals. Each animal type is an OTU. Consider the set of OTUs shown below. They have been scored for dichotomous characters (1 = present, 0 = absent).

OTU	Characters			
	Hair	Lungs	Egg-laying	Milk
Dog	1	1	0	1
Turtle	0	1	1	0
Canary	0	1	1	0
Goldfish	0	0	1	0

One measure of similarity between OTUs i and j employs the numbers of matches and mismatches between their character states. In this case, there are four characters, and the states for each character may be 0 or 1. The matches and mismatches are conveniently represented in a table:

		OTU j	
		1	0
OTU i	1	a	b
	0	c	d

In this table,

- a = the number of characters for which OTUs i and j are both 1,
- b = the number of characters for which OTU i is 1 and OTU j is 0,
- c = the number of characters for which OTU i is 0 and OTU j is 1,
- d = the number of characters for which OTUs i and j are both 0.

Note that the number of characters equals $a+b+c+d$. The **simple matching coefficient** is defined as

$$s_{ij} = (a + d)/(a + b + c + d).$$

For the particular comparison of the canary and goldfish, the table becomes

		canary	
		1	0
goldfish	1	1	0
	0	1	2

and $s_{ij} = (1 + 2)/4 = 0.75$. All of the s_{ij} gathered together in a matrix constitute a **similarity matrix**. Such a matrix is $m \times m$ (where m is the number of OTUs) and symmetric.

Sometimes negative matches convey no additional information about relationships. For instance, including feathers as a character would cause the value of d for pairwise comparisons among dogs, goldfish, and turtles to increase. (Any two of the three would both score 0 relative to this character.) Does it really make sense to include an additional character for which all OTUs score 0 if only these three animals are to be compared? For this reason, sometimes it is preferable to define similarity in terms of **Jaccard's coefficient**,

$$s_{ij} = \frac{a}{a + b + c}.$$

This similarity coefficient only “counts” characters that are present or that differ between the two OTUs.

It is sometimes more useful to compare OTUs in terms of **dissimilarities** rather than in terms of similarities. The dissimilarity d_{ij} corresponding to Jaccard's coefficient is

$$d_{ij} = \frac{b + c}{a + b + c} = \frac{a + b + c - a}{a + b + c} = 1 - s_{ij}.$$

10.3.1 Dissimilarities and Distances Measured on Continuous Scales

There are desirable properties that dissimilarities d_{ij} may have, and when they have them they are called **distances** or **metrics**. These properties are:

1. Symmetry: $d_{ij} = d_{ji}$ for each i, j ;
2. Distinguishability: $d_{ij} \neq 0$ if, and only if, $i \neq j$;
3. **Triangle inequality**: $d_{ij} \leq d_{ik} + d_{kj}$ for each i, j, k .

An example of such a metric or distance is the familiar **Euclidean distance** in two dimensions. If OTU i has character values (x_{i1}, x_{i2}) and OTU j has character values (x_{j1}, x_{j2}) , then this distance is given by

$$d_{ij} = \sqrt{(x_{i1} - x_{j1})^2 + (x_{i2} - x_{j2})^2}. \quad (10.1)$$

This is illustrated in Fig. 10.1A.

In general, we are concerned with many characters (possibly more than there are letters in the Roman alphabet). Extending our earlier notation, the values of the p characters for OTU i are denoted by $x_{i1}, x_{i2}, \dots, x_{ip}$. In this notation, the first subscript is the OTU or object label and the second subscript corresponds to the particular character. We can organize the data into a matrix with m rows of OTUs and p columns of characters. The Euclidean distance between OTUs i and j is then defined by

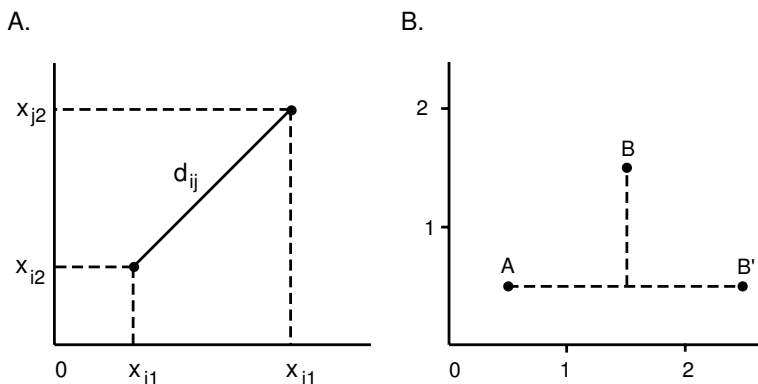


Fig. 10.1. Illustration of the Euclidean distance between OTU i and OTU j having two characters (panel A) and of the city-block distance (panel B). In panel B, the labels A, B, and B' represent different OTUs.

$$d_{ij} = \sqrt{\sum_{k=1}^p (x_{jk} - x_{ik})^2}.$$

An alternative to the Euclidean distance metric is the **city-block metric** (sometimes called the “Manhattan” distance), which is defined as

$$d_{ij} = \sum_{k=1}^p |x_{ik} - x_{jk}|.$$

This measure is analogous to the way that we would measure distances if we were driving between points in a city laid out in blocks. (We must travel along the city’s street grid and not the direct route “as the crow flies”.) This means that if OTU i and OTU j are 2 units apart with respect to character 1, they are as far apart as if they were 1 unit apart with respect to character 1 and 1 unit apart with respect to character 2 (Fig. 10.1B). If the Euclidean distance were used in the latter case, the distance would be $\sqrt{2}$.

The city-block metric makes sense for some applications. Suppose that we are comparing the following amino acid sequences:

A ... P R H L Q L A V R N ...	A ... P R H L Q L A V R N ...
B ... P R H V L L A V R N ...	B' ... P R H A Q L A V R N ...
0 0 0 1 1 0 0 0 0	0 0 0 2 0 0 0 0 0

Below each pair is the number of mutations in the nucleic acid sequence underlying A needed to produce B or B'. In the case on the left, there have been two mutations, one at each of two different positions. In terms of mutations, B and B' are equally distant from A: two mutations away. In the case on the right, there have been two successive mutations at the same position. We

would not want to say that the distance between B and A is $\sqrt{2}$, while the distance between B' and A is 2.

The generalization of the distance metrics given above is the Minkowski metric:

$$d_{ij} = \left[\sum_{k=1}^p |x_{ik} - x_{jk}|^a \right]^{1/a}.$$

Clearly, the city-block metric corresponds to $a = 1$ and the Euclidean distance corresponds to $a = 2$.

10.3.2 Scaling Continuous Character Values

In many applications, such as measuring mRNA expression levels with microarrays, the character values are drawn from continuous distributions of possible values. In addition, different characters may not have the same units. For example, in classifying fossil hominids, we might use height (in meters), molar surface area (mm^2), and cranial volume (cm^3). Clearly, trying to calculate Euclidean distance when the coordinates (characters) have different units is problematic. By scaling coordinate values, they all become dimensionless and this problem disappears.

Let x_{ik} represent the value of character k corresponding to OTU i . The standardized or scaled character values might be written as

$$x_{ik}^* = \frac{x_{ik}}{s_k}, \quad k = 1, 2, \dots, p,$$

where s_k is the standard deviation of character k measured over all OTUs. Division by s_k ensures that x_{ik}^* is dimensionless. The other important result of this division is that it adjusts for characters or coordinates that have much greater variation than the others. Such variation, if not compensated for by scaling, would give those coordinates undue weight when distances are calculated—coordinates with the broadest range would dominate the analysis. For the fossil hominid example, it would be silly to have height contribute more just because it was stated in centimeters rather than meters. The scaling above eliminates this problem.

The scaling described above was for each *column* of characters. Sometimes (for example, in microarray experiments) the set of characters for each object are all measured on the same dimensionless scale. The characters for each gene in a microarray experiment might correspond to a time series for which the expression ratios as a function of time are measured. The actual amplitude of the measurement at any time point may be less important than the *pattern* of values for all points taken as a whole. In this case, the scaling should be over all time points (characters) for each gene (object). In other words, the scaling is applied to rows rather than columns.

10.4 Clustering

There are two general approaches to clustering. **Hierarchical clustering** consists of the successive joining together or splitting of individual objects or groups of objects based upon a measure of similarity or distance between the objects. This produces a “tree” or dendrogram in which any object is associated with successively larger groups of objects. Groups or clusters are arbitrarily defined based upon distance relationships to be described. This hierarchical structure resembles the hierarchical patterns of phylogenetic relationships among organisms, which are a consequence of their evolutionary descent from shared common ancestors. The second clustering approach involves *optimization*, in which the number of groups or clusters is prespecified. Optimization (nonhierarchical) methods do not construct the clusters based on pairwise differences between individual objects. Instead, mutually exclusive clusters are formed, with no “subclusters.” We discuss *K*-means as an example of this. Assignments of objects to one or another of the predetermined groups are determined in such a way that their distances from cluster “centers” are minimized. A method such as *K*-means is not used for classification of organisms because the biology and observation indicate that their relationships are hierarchical.

10.4.1 Agglomerative Hierarchical Clustering

Agglomerative hierarchical methods sequentially merge OTUs into clusters having larger and larger numbers of members by successively grouping those that are least distant first. In contrast, divisive hierarchical methods sequentially divide larger groups into smaller and smaller clusters with fewer members. Here, we will discuss an agglomerative approach only. Agglomerative hierarchical clustering starts with the $m \times p$ matrix of m OTUs measured on p characters. This is used to calculate an $m \times m$ **distance matrix** using distance measures such as those described above. The distance matrix is used to decide which of the OTUs to join first. After the first two OTUs are joined, a new distance matrix is generated taking into account the newly formed cluster, and the process is continued until all OTUs have been joined to other OTUs or to clusters. The result is presented as a **dendrogram**. The process is summarized in the following steps.

Procedures for agglomerative clustering

1. Start with m clusters, each containing one OTU, and calculate the $m \times m$ symmetric distance matrix D_1 (entries d_{ij}).
2. Determine from D_1 which of the OTUs (or clusters in later iterations) are least distant (i.e., find the i and j for which d_{ij} is a minimum, $i \neq j$). Suppose these happen to be clusters (or OTUs) I and J.

3. Merge I and J into cluster IJ. Form a new distance matrix D_2 by deleting rows corresponding to I and J and columns I and J, and by adding a new row and column for distances of IJ from all remaining clusters.
(The method for calculating distances between clusters is given in (10.2).)
4. Repeat steps 2 and 3 a total of $m - 1$ times until a single cluster is formed.
 - Record which clusters are merged at each step.
 - Record the distances between the clusters that are merged in that step.

As the steps above are applied, individual OTUs merge with others to form clusters, and it is necessary to specify what is meant by distance between clusters. There are a number of possible criteria for merging clusters I and J. Three clustering criteria, based upon the individual OTUs that belong to I and J, are

$$\begin{aligned}
 \text{Single linkage:} & \quad d_{IJ} = \min\{d_{ij} : i \in I \text{ and } j \in J\}, \\
 \text{Complete linkage:} & \quad d_{IJ} = \max\{d_{ij} : i \in I \text{ and } j \in J\}, \\
 \text{Group average:} & \quad d_{IJ} = \sum_{i \in I} \sum_{j \in J} d_{ij} / (N_I N_J),
 \end{aligned} \tag{10.2}$$

where N_I and N_J are the numbers of members in clusters I and J, respectively. $N_I N_J$ is the number of possible distances between each of the objects in cluster I and the objects in cluster J. These distinctions are illustrated in Fig. 10.2.

This method is illustrated by using the primate data presented in Table 10.1. In that table, the fraction of bases that differed between each sequence pair was stated. This edit distance is used to produce our distances, the d_{ij} . The table of distances between OTUs is given below. Because $d_{ij} = d_{ji}$, we need only list half of the off-diagonal entries. For the entries on the diagonal, $d_{ii} = 0$. (The distance between an OTU and itself is zero.) The matrix below represents the initial distance matrix, D_1 .

	OTU	Hy	Pa	Go	Ho
$D_1 =$	Hy	0			
	Pa	0.150	0		
	Go	0.150	0.133	0	
	Ho	0.150	0.067	0.100	0

For illustration, we employ single-linkage clustering, for which the first cluster is determined by $\min\{d_{ij}\}$, $i \neq j$. Clearly, Pa (*Pan troglodytes*, the chimpanzee) and Ho (*H. sapiens*) have the least pairwise distance: $d_{\text{Pa, Ho}} = 0.067$ (indicated by the box). So we merge these two OTUs to form the first cluster (PaHo), and we record the joining of Pa and Ho at distance 0.067. We then create a new distance or dissimilarity matrix, D_2 , by deleting rows and columns corresponding to Pa and Ho and adding a new row and column for the (PaHo) cluster.

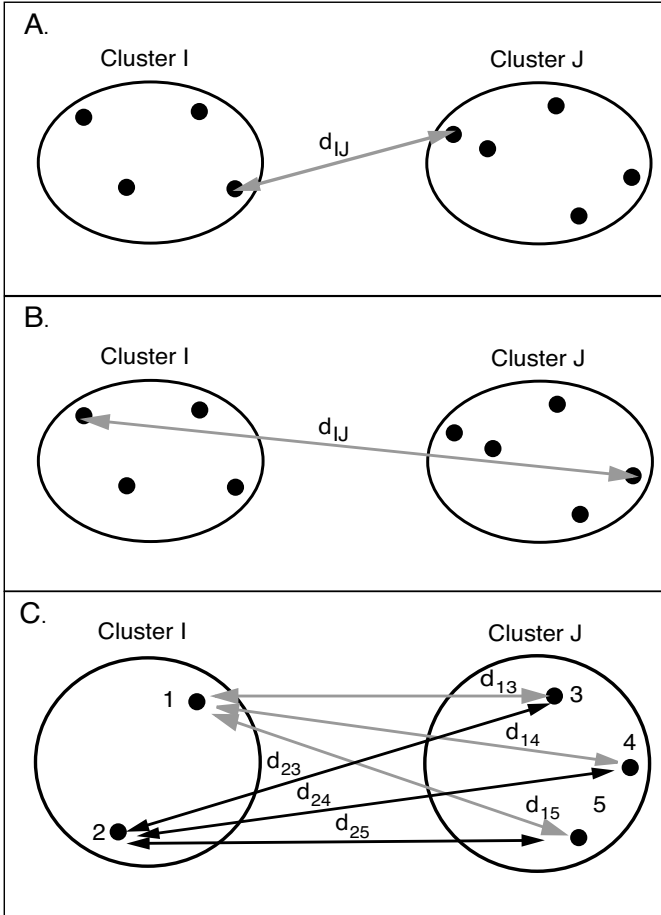


Fig. 10.2. Methods of defining the distance between clusters. Panel A illustrates single linkage, panel B complete linkage, and panel C group average linkage.

$$D_2 = \begin{array}{c|cc} \text{OTU} & \text{Hy} & \text{Go (PaHo)} \\ \hline \text{Hy} & 0 & \\ \text{Go} & 0.150 & 0 \\ \text{(PaHo)} & 0.150 & \boxed{0.100} & 0 \end{array}$$

The entries in the last row were calculated from the data in D_1 as $d_{(\text{PaHo})W} = \min\{d_{\text{Pa}W}, d_{\text{Ho}W}\}$, where W is either Hy or Go . Thus $d_{\text{Hy}(\text{PaHo})} = \min\{d_{\text{HyPa}}, d_{\text{HyHo}}\} = 0.15$, and $d_{\text{Go}(\text{PaHo})} = \min\{d_{\text{GoPa}}, d_{\text{GoHo}}\} = 0.10$. The next cluster to form is determined by the minimum entry in the distance matrix D_2 , and this corresponds to forming the cluster $\text{Go}(\text{PaHo})$ at distance 0.10. For the next iteration, we create distance matrix D_3 ,

$$D_3 = \begin{array}{c|cc} \text{OTU} & \text{Hy} & \text{Go(PaHo)} \\ \hline \text{Hy} & 0 & \\ \text{Go(PaHo)} & \boxed{0.15} & 0 \end{array}$$

where the distance between Hy and the Go(PaHo) cluster is $\min\{d_{\text{HyPa}}, d_{\text{HyHo}}, d_{\text{HyGo}}\}$ ($=\min\{0.15, 0.15, 0.15\}$ from matrix D_1). This merges the last OTUs with the others. The dendrogram representing this clustering is shown in Fig. 10.3.

Computational Example 10.1: Hierarchical clustering using R

This box illustrates how to perform cluster analysis using R. We use the primate data once more. The first task is to make a file of the sequence data shown in Table 10.1; as above, we use just the first four species. The file `primates.txt` has 4 rows and 60 columns, coded as $a = 1$, $c = 2$, $g = 3$, $t = 4$. Having read in the data using

```
> seqs<-matrix(scan("primates.txt"),nrow=4,byrow=T)
```

the next task is to compute the distance matrix. In this example, the distance between two sequences is the proportion of positions at which they differ. The following function evaluates this distance matrix:

```
seqdist<-function(x,n)
{
  dmat<-matrix(nrow=n,ncol=n)
  for(i in 1:n){
    for(j in 1:n){
      dmat[i,j]<-length(x[j,][x[j,]!=x[i,]])/length(x[1,])
    }
  }
  return(dmat)
}

> dapes<-seqdist(seqs,4)
> dapes
      [,1]      [,2]      [,3]      [,4]
[1,] 0.00 0.15000000 0.1500000 0.15000000
[2,] 0.15 0.00000000 0.1333333 0.06666667
[3,] 0.15 0.13333333 0.0000000 0.10000000
[4,] 0.15 0.06666667 0.1000000 0.00000000
```

We note that other distances may be calculated using the R function `dist`. The remainder of the clustering is straightforward. For single-linkage clustering, it results in the dendrogram in Fig. 10.3.

```
> # make dist object from the data
> dapes1<-as.dist(dapes, diag=FALSE, upper=FALSE)
> #add species labels
> species=c("Hy","Pa","Go","Ho")
```

```
> p1clust(hclust(dapes1,"single"),labels=species,xlab="",
+ sub="")
```

It is possible to draw the dendrogram with all the leaves extending to the same level (it has the same meaning: the distances at which OTUs join are the relevant data). Try

```
> p1clust(hclust(dapes1,"single"),labels=species,xlab="",
+ sub="",hang=-1)
```

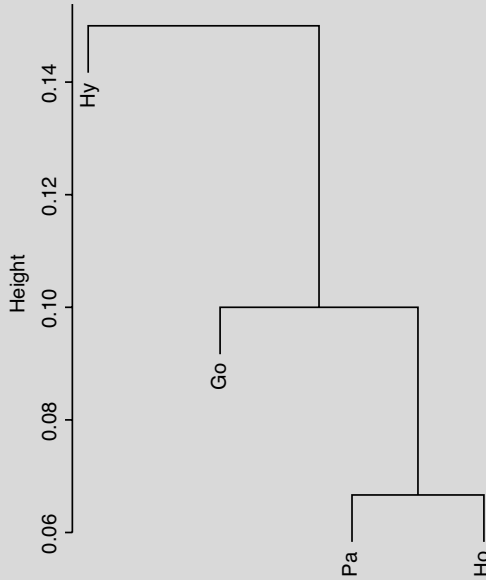


Fig. 10.3. Dendrogram of single-linkage clustering for primate species.

10.4.2 Interpretations and Limitations of Hierarchical Clustering

The example given above is very simple. If we were clustering a much larger number of OTUs, the question of *number of clusters* will arise. Consider the dendrogram shown in Fig. 10.4. It seems natural to think that there are three clusters: AB, CD, and EF. This impression is based upon the long distances separating these clusters from each other and the observation that each of the pairs (C and D, for example) are less distant from each other than either of them is from any other OTU. However, we should recognize that the distance criterion that we use to define the clusters will determine the number of clusters assigned. If we set the criterion at distances between 0.4 and 0.3, we would define two clusters. If instead we focused on distances between 0.3

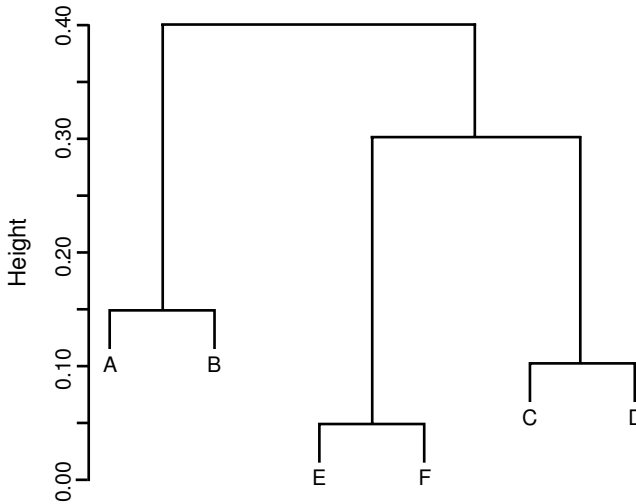


Fig. 10.4. Dendrogram illustrating numbers of clusters for different distance criteria.

and 0.15, we would define three clusters; the number of clusters is somewhat arbitrary.

Another issue (mentioned in the previous section) is the reliability of the particular clustering that has been produced by the procedure. The hierarchical method has a number of limitations:

1. The choice of distance measure is important. Methods for incorporating an evolutionary model into the comparison of DNA sequences are discussed in Chapter 12.
2. There is no provision for reassigning objects that have been incorrectly grouped. The method proceeds in a bottom-up fashion and, once joined to a cluster, an OTU remains at that position in the hierarchy.
3. Errors are not handled explicitly in the procedure.
4. No single method of calculating intercluster distances is universally the best. We need to test the results by using different definitions of intercluster distance. In some “head-to-head” tests, single-linkage clustering is reportedly least successful and group average clustering tends to do fairly well. For a review of the problem, see Everitt and Dunn (2001).
5. Single-linkage clustering may tend to join clusters whose centroids are fairly distant if these clusters are joined by a chain of OTUs.

It is important to know how *robust* the hierarchical method is in any particular application. (A statistical method is called robust if it works well for

a variety of input data and initial parameters.) Reliable data analysis requires some knowledge of the magnitude experimental error. Operationally, we can test the topology of the clusters obtained for any particular calculation by repeating the clustering several times after adding appropriate random errors to coordinates of all OTUs. If the same groupings of OTUs into clusters is achieved consistently after repeatedly adding different sets of random errors, then the solution to the clustering problem is said to be *stable*. Another test of the method with m OTUs is to repeat the analysis m times for $m - 1$ OTUs, leaving out a different OTU each time. (This is sometimes called a “jackknife” procedure since we are “flipping out” one of the OTUs like a jackknife blade.) If the clusters of other OTUs change dramatically with the omission of one OTU, then we might doubt the reliability of clustering based on that data set.

The dendrograms resemble phylogenetic trees, which are presented in the next chapter. A phylogenetic tree is a hypothesis about the genealogical relationships between organisms based upon shared common ancestors. Dendrograms represent the joining of OTUs or clusters based upon pairwise distances (defined as edit distances in our example) using a stated clustering method. Except for stating that changes at position i are independent from changes at position j in the DNA sequence, there was no biology built into the example above. For phylogenetic trees, it is assumed that each of the two bifurcating branches represents a trajectory of independent evolution. Accordingly, distances between two OTUs are apportioned between the two branches (i.e., the sum of the two branch lengths is proportional to the distance between OTUs). The branch lengths may be proportional to the number of mutational events from any node. In phylogenetic trees, internal nodes will correspond to ancestors having a set of hypothetical character states. As we will see later, there are assumptions about nucleotide substitutions and reversions and the rates at which these occur along each branch. Building phylogenetic trees is procedurally similar to clustering but with biological models included.

10.5 *K*-means

We use *K*-means (Hartigan and Wong, 1979) as an example of an optimization method. As we will see, many iterations of the procedure may be required, so a statistical software package is necessary for anything except “toy” problems. *K*-means is a non-hierarchical clustering method that involves prior specification of the number of clusters, k . It does not require prior computation of a pairwise distance matrix, because the relevant distance is the distance of each OTU from the cluster center, or **centroid**. The rationale is that OTUs will be allocated into the k clusters such that the within-cluster sums of squares of distances from cluster centroids (within-ss), summed over all clusters, will be minimized.

$$\begin{aligned} \text{within-ss, cluster } j &= \sum_i d_{ij}^2, \quad i = 1, \dots, m_j, \\ \text{total within-ss } S &= \sum_j \sum_i d_{ij}^2, \quad i = 1, \dots, m_j; j = 1, \dots, k. \end{aligned}$$

This is illustrated in Fig. 10.5 for two alternative values of k : $k = 2$ and $k = 3$. Increasing k from 2 to 3 splits the larger cluster at the bottom into two clusters, leaving the one at the top unchanged. The K -means calculation involves trying different centroid positions and iteratively testing each OTU for its distance to one of the centroids. Each OTU is assigned membership to the cluster whose centroid is closest. Once the cluster memberships are assigned after each iteration, new centroid positions are calculated, and the process is repeated.

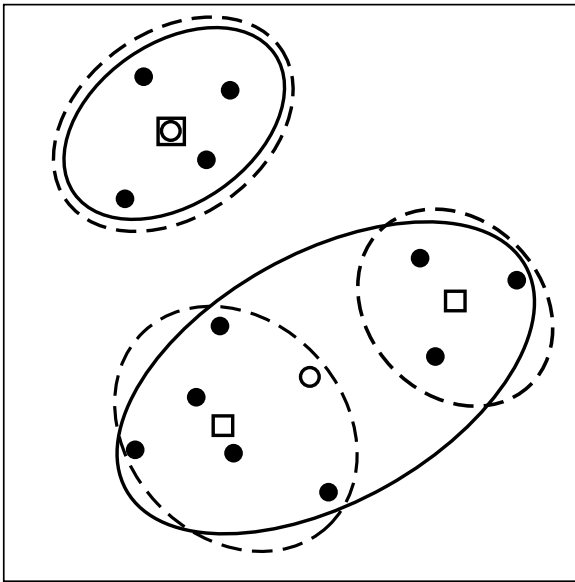


Fig. 10.5. K -means clustering of 12 objects (filled circles) having two characters. Clusters for $k = 2$ (solid lines) and $k = 3$ (dashed lines) are indicated. Cluster centers for $k = 2$ or $k = 3$ are denoted by open circles or squares, respectively.

It is prudent to plot the data for a visual “reality check.” This is easy if there are just two dimensions, but if there are many dimensions, we will need to look at projections of the data on planes defined by coordinate pairs. It is also useful to repeat the process with different partitions or initial group centers. This is because some choices of initial partitioning may be bad, and results may be affected by outliers (OTUs whose coordinates are very different from those of other OTUs). We should also repeat the process for different val-

ues of k , unless we have prior information indicating the appropriate number of clusters.

Procedures for K -means clustering

1. Partition the OTUs into k clusters. (This can be done by random partitioning or by arbitrarily clustering around two or more OTUs.)
2. Calculate the centroids of the clusters.
3. Assign or reassign each OTU to that cluster whose centroid is the closest. (Distance is calculated as Euclidean distance.)
4. Recalculate the centroids of the new clusters formed after the gain or loss of OTUs to or from the previous clusters.
5. Repeat steps 3 and 4 for a predetermined number of iterations or until membership of the groups no longer changes.

As a first example of how this works, we will use a “toy” problem that can be calculated by hand. We imagine that we have five OTUs, A, B, . . . , E, each described by two characters, x_1 and x_2 , and we want to place the five objects into two clusters. The data are:

OTU	x_1	x_2
A	1	1
B	3	1
C	4	8
D	8	10
E	9	6

The data are plotted in Fig. 10.6. We can already see by inspection what the answer should be (A and B form one cluster, while C, D, and E form another), but we want to get a feel for how the computation actually works.

Computational Example 10.2: K -means clustering by hand

Step 1. Start by making an arbitrary partition of OTUs into clusters: OTUs with $x_1 \leq 6$ will be taken as Cluster I and the others will be taken as Cluster II. Thus we assign A, B, and C to Cluster I, and D and E to Cluster II.

Step 2. Calculate the centroids for the first definition of clusters. For Cluster I (defined as A, B, C), this value is $x_1 = 2.67$, $x_2 = 3.33$, while for Cluster II (defined as D, E), the centroid is $x_1 = 8.50$, $x_2 = 8.00$.

Step 3. Now calculate the Euclidean distance between each OTU and each of the two cluster centroids:

$$\begin{aligned}
 d(\text{A, I}) &= 2.87 & d(\text{A, II}) &= 10.26 \\
 d(\text{B, I}) &= 2.35 & d(\text{B, II}) &= 8.90 \\
 d(\text{C, I}) &= 4.86 & d(\text{C, II}) &= 4.50
 \end{aligned}$$

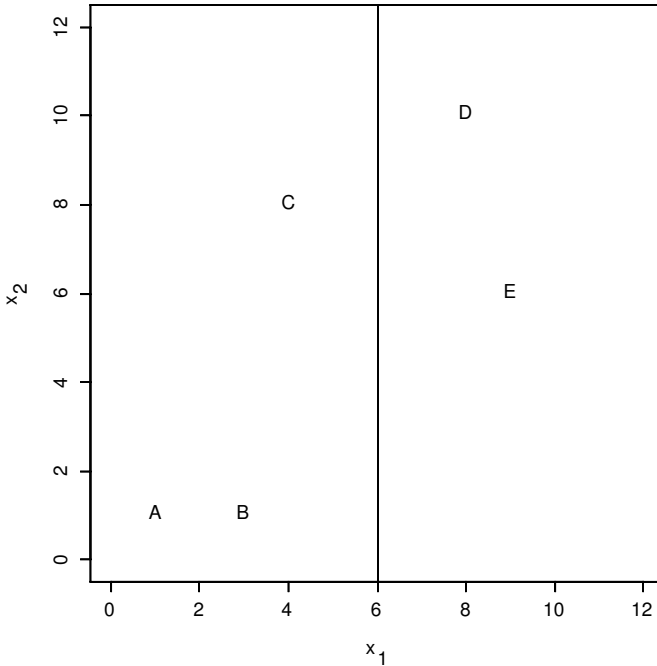


Fig. 10.6. Plot of data and initial partitioning for a simple hand-calculated example of *K*-means clustering. For details, see the text.

$$\begin{aligned} d(D, I) &= 8.54 & d(D, II) &= 2.06 \\ d(E, I) &= 6.87 & d(E, II) &= 2.06 \end{aligned}$$

Step 4. A and B are closer to the centroid of Cluster I than they are to the centroid of Cluster II, and D and E are closer to the centroid of Cluster II than they are to the centroid of Cluster I. However, C is closer to the centroid of Cluster II. Therefore, reassign C to Cluster II. Other OTUs retain their previous assignments.

Step 5. Calculate centroids for Cluster I' composed of A and B and Cluster II' composed of C, D, and E. For Cluster I' , this new centroid is $x_1 = 2.00$, $x_2 = 1.00$, while for Cluster II' the new centroid is $x_1 = 7.00$, $x_2 = 8.00$.

$d(A, I') = 1.00$	$d(A, II') = 9.22$
$d(B, I') = 1.00$	$d(B, II') = 8.06$
$d(C, I') = 7.28$	$d(C, II') = 3.00$
$d(D, I') = 10.82$	$d(D, II') = 2.24$
$d(E, I') = 8.60$	$d(E, II') = 2.83$

Since A and B are closer to the centroid of Cluster I' than to that of Cluster II' , no reassignment of cluster membership is needed. Similarly, since

C, D, and E are all closer to the centroid of Cluster II' than to the centroid of Cluster I', no reassignment of cluster membership is needed for them either. The assignments shown in the boxes above are the final result, which agrees with our “eyeball” estimate.

As a second example, consider the following living and fossil hominoid species, each described by two characters: brain mass and body mass. (*H.* = *Homo*, *A.* = *Australopithecus*, early and late *H. erectus* designated by E and L.)

RAW DATA			SCALED DATA		
	body mass (kg)	brain mass (g)		body mass	brain mass
<i>H. sapiens</i>	53	1355	<i>H. sapiens</i>	2.708	4.426
<i>H. erectus</i> L	57	1016	<i>H. erectus</i> L	2.913	3.318
<i>H. erectus</i> E	55	804	<i>H. erectus</i> E	2.811	2.626
<i>H. ergaster</i>	58	854	<i>H. ergaster</i>	2.964	2.789
<i>H. habilis</i>	42	597	<i>H. habilis</i>	2.147	1.950
<i>A. robustus</i>	36	502	<i>A. robustus</i>	1.840	1.640
<i>A. boisei</i>	44	488	<i>A. boisei</i>	2.249	1.594
<i>A. africanus</i>	36	457	<i>A. africanus</i>	1.840	1.493
<i>A. afarensis</i>	37	384	<i>A. afarensis</i>	1.891	1.254
<i>P. troglodytes</i>	45	395	<i>P. troglodytes</i>	2.300	1.290
<i>G. gorilla</i>	105	505	<i>G. gorilla</i>	5.366	1.649

The first step is to scale the data by dividing each value in each column of the raw data by the standard deviation of the entries in that column. Notice how scaling causes the ranges of the data in the two columns to become more nearly the same. The effect of this is that both of the variables, brain mass and body mass, will be weighted approximately equally in the clustering process. Because *K*-means clustering in this example is far beyond what is reasonable to attempt with a hand calculator or spreadsheet application, we provide the R commands to perform it in Computational Example 10.3.

Computational Example 10.3: *K*-means clustering using R

This box illustrates the use of R for *K*-means clustering. We use the primate data from Section 10.5. First, we import the data from a file and scale them:

```
> raw.dat<-read.table("Raw_Data")
> scaled.dat<-raw.dat
#Divide column entries by respective column SDs for scaling
> scaled.dat[,1]<-raw.dat[,1]/sqrt(var(raw.dat[,1]))
> scaled.dat[,2]<-raw.dat[,2]/sqrt(var(raw.dat[,2]))
```


Next, we perform *K*-means clustering, with $k = 2$. We could either specify the number of clusters or provide a 2×2 matrix identifying initial cluster centers. We use the latter approach, dividing the data into two groups by a horizontal line h midway between the extreme values of scaled brain mass. We assign the average of all values of scaled body mass (`scaled.dat[,1]`) to the x values for the initial two cluster centers. The y value for one of the initial cluster centers is assigned the mean value of scaled brain mass (`scaled.dat[,2]`) less than h . The y value for the other initial cluster center is taken as the average value of scaled brain mass less greater than h .

```
> body<-scaled.dat[,1]
> brain<-scaled.dat[,2]
> h<-mean(c(4.425578,1.254186)) #Average of extreme values
> h
[1] 2.839882 #Horizontal line h(x)=2.839882
> x1<-x2<-mean(body) #x values for both initial clusters
> y1<-mean(brain[brain<h]) #y value for initial cluster 1
> y2<-mean(brain[brain>h]) #y value for initial cluster 2
> in.cent<-cbind(c(x1,x2),c(y1,y2))
#Matrix of coordinates for initial centers
> in.cent
      [,1]      [,2]
[1,] 2.638996 1.809424
[2,] 2.638996 3.871972
```

We apply *K*-means with $k = 2$ using `in.cent`, the matrix of coordinates for initial cluster centers:

```
> k.dat2<-kmeans(scaled.dat,in.cent,iter.max=10)
> k.dat2
#Cluster to which OTUs 1:11 each belongs
$cluster
 [1] 2 2 2 2 1 1 1 1 1 1 2
$centers #Coordinates of final cluster centers
      [,1]      [,2]
1 2.044293 1.536704
2 3.352640 2.961708
$withinss
 [1] 0.5517516 9.2416832
$size #Number of members in each cluster
 [1] 6 5
```

With $k = 2$, we find two clusters, one of which includes the first four species of *Homo* together with *Gorilla*. Notice how the coordinates of the final cluster centers have changed compared to the initial values. Note also that *Gorilla* has a scaled brain mass of 1.649, which means that it originally belonged to

cluster 1. After K -means, it has moved up to cluster 2. We try again with $k = 3$, this time just specifying the number of clusters rather than the initial cluster centers:

```
> k.dat3<-kmeans(scaled.dat,3,iter.max=10)
> k.dat3
$cluster
 [1] 3 3 3 3 2 2 2 2 2 2 1
$centers
      [,1]      [,2]
1 5.366268 1.649385
2 2.044293 1.536704
3 2.849233 3.289788
$withinss
 [1] 0.0000000 0.5517516 2.0205711
$size
 [1] 1 6 4
```

This is much better, based on reduction of within-cluster sums of squares. A plot of the data is shown in Fig. 10.7.

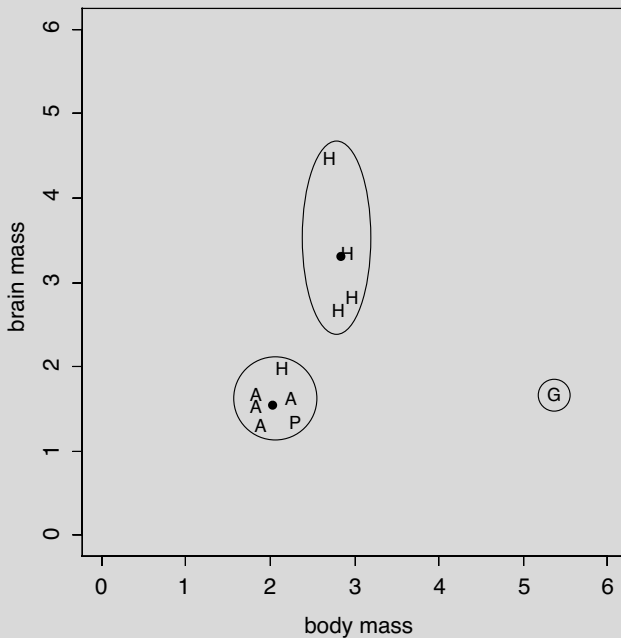


Fig. 10.7. K -means clustering of hominoids based upon scaled brain mass and body mass. The genera *Homo*, *Australopithecus*, *Pan*, and *Gorilla* are indicated by H, A, P, and G, respectively. The three clusters for $k = 3$ are enclosed by ellipses or circles, and cluster centers for multimember clusters are indicated by filled circles.

Gorilla now forms a cluster by itself. In the other two clusters, all members of genus *Homo* are indicated by H. We can see that all but one of these form a separate cluster. On the basis of these characters, *H. habilis* is clustered with *Australopithecus* species. (Analysis using many characters has led some anthropologists to reclassify *H. habilis* as *A. habilis*.)

How many clusters should there be? This number is, to some extent, arbitrary. If we increase k until it equals the number of OTUs, it is possible to make the within-ss zero! But then we are back where we started: a set of m unclustered OTUs. One way to decide the appropriate number of clusters is to repeat the process for several different values of k and plot the sum S of within-ss values (for all clusters) as a function of k . As k increases, we will observe that S decreases. It is probably not prudent to increase k beyond the point at which S is showing large decreases. This is illustrated in Fig. 10.8 for the scaled hominoid data. Note that increasing k from 2 to 3 produced a large drop in S , while going from $k = 3$ to $k = 4$ produced a much more modest decrease by splitting off a single-member group. We conclude that $k = 3$ is an appropriate choice, consistent with the appearance of the data in Fig. 10.7.

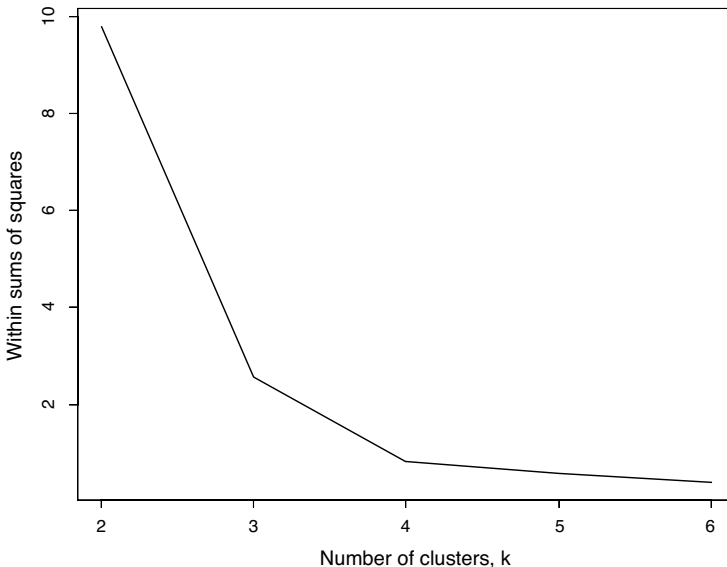


Fig. 10.8. Response of within-cluster sums of squares (summed over all clusters) to increasing values of k . Data are from clustering of hominoids (results for $k = 3$ shown in Fig. 10.7). Substantial reduction in within-ss occurs when k is increased from 2 to 3. Further reductions in within-ss result for values of k larger than 3, but improvements are comparatively small, and additional single-member clusters are produced.

10.6 Classification

We briefly touch upon the classification process, leaving the details to more advanced studies. We applied the classification process to biological signals in the last chapter. In that case, we determined a distribution of scores for bona fide signals and another distribution of scores for the “background” sequence. Classification as signal or nonsignal was based on how the score of any sequence compared with a cutoff score, with scores defined in terms of log-odds ratios. Assignment of a signal to one or the other class was made, with the sensitivity and specificity of the assignment determined by the score distributions.

In the context of this chapter, we imagine that we are provided with an OTU, X , not employed in creating the clusters, and we ask to which of the clusters that have already been constructed X belongs. For K -means, this is quite straightforward: we only need to calculate the Euclidean distance from X to the centroids of each of the k clusters and then assign X to the closest one. For hierarchical clustering, it is common to add X to all other OTUs and repeat the clustering. If X falls within one of the previously defined clusters, and branches of all other defined clusters in the dendrogram are connected as before, then its membership in a previously defined cluster is established. However, it may be that the addition of X changes the branching order of OTUs from two or more clusters (as might happen if they were all very similar to start with). This would suggest that the original dendrogram was not very robust. (The application uses the same principle as the jackknife procedure mentioned above.)

References

- Dunn G, Everitt BS (1982) *Introduction to Mathematical Taxonomy*. Cambridge: Cambridge University Press.
- Everitt BS, Dunn G (2001) *Applied Multivariate Data Analysis* (2nd edition). Oxford: Oxford University Press.
- Hartigan J, Wong M (1979) A K -means clustering algorithm. *Applied Statistics*, 28:100–108.
- Johnson RA, Wichern DW (2002) *Applied Multivariate Statistical Analysis*. Englewood Cliffs, NJ: Prentice-Hall.

Exercises

Exercise 1. Present an expression for dissimilarity d_{ij} and its relationship to s_{ij} when s_{ij} is the *simple matching coefficient*.

Exercise 2. Show whether or not distances computed using $z_{ik} = (x_{ik} - \mu_k)/s_k$, where μ_k is the mean of character k , will differ from distances computed from x_{ik}^* (Section 10.3.2).

Exercise 3. Plot the scaled data for *H. habilis*, *A. robustus*, *A. boisei*, *A. africanus*, *A. afarensis*, and *P. troglodytes* presented in Section 10.5 (the lower left-hand cluster in Fig. 10.7). Use a ruler to measure distances, and sketch what you think the hierarchical relationships would be from hierarchical clustering. [Hint: You can check your answer from the results of Exercise 6 below.]

Exercise 4. This exercise explores the consequences of having different characters highly correlated. Suppose that five OTUs A, B, . . . , E are being clustered based upon five characters x_1, x_2, x_3, x_4 , and x_5 . The matrix of characters, X , is

	x_1	x_2	x_3	x_4	x_5
A	1	1	2	1	3
B	2	3	2	3	6
C	6	8	4	7	2
D	8	9	4	3	1
E	6	6	3	9	9

- Perform hierarchical clustering of these five OTUs, and note the topology of the dendrogram.
- Determine the correlation coefficients for each pair of variables, and represent values for highly correlated characters by a single column vector (combining values for each OTU using a statistic of your choice).
- Now use the composite character defined in part b and the remaining columns from matrix X to form a new matrix having a lesser number of characters, and again perform hierarchical clustering. Compare the topology of the dendrogram to the one observed in part a.
- Criticize or defend the approach used in part c.

Exercise 5. With the aid of R, perform hierarchical clustering for the following OTUs measured on characters x and y . For `dist` use `method="euclidean"` and for `hclust` use `method="complete"`. Note with which OTUs C clusters.

	x	y
A	1.0	9.0
B	2.0	8.0
C	4.0	6.0
D	5.5	4.5
E	6.5	3.5
F	8.0	2.0

Now repeat the process for all OTUs except D, and observe the cluster membership of C. Perform a scatter plot for OTUs A, . . . , F using the `plot` and

points functions, and explain why C changed its group membership when D was omitted.

Note: OTUs arranged like this can create other problems when single-linkage clustering is used: see Section 10.4.2 and Everitt and Dunn (2001) for a discussion of *chaining*.

Exercise 6. Use K -means to cluster the data in Exercise 5, $k = 2$, first including all data and then including all OTUs except for D.

- Does the cluster membership of C change when D is excluded?
- Try to force OTU C into a cluster with D, E, and F by setting the initial cluster centers at the coordinates of A and D.

Exercise 7. To test the effects of different measures of intercluster distance, perform hierarchical clustering on the following OTUs:

	x	y
A	1.0	1.5
B	1.0	1.0
C	3.0	1.0
D	5.5	1.0
E	7.0	1.0
F	7.0	2.0
G	7.0	5.0
H	6.0	6.0
I	8.0	6.0

- First perform hierarchical clustering using single-linkage clustering. How many clusters having three closely related members are there? Which two clusters are more closely related?
- Repeat the hierarchical clustering, except this time use complete linkage clustering. How many three-member clusters are there? Which two of these three-member clusters are more closely related?
- Plot the data points, and explain the reasons why a and b above gave different results. Also, explain with a diagram why the G, H, I clusters differ for single-linkage and complete-linkage clustering.

Exercise 8. Perform hierarchical clustering using all of the data at the top of Table 10.1 (see Computational Example 10.1). For characters, first produce a string 60 characters long that represents the consensus sequence for the regions shown. Then, for each OTU, score each position that agrees with this consensus 1 and each position that disagrees 0. This character table should be used as input to the R function `dist` using `manhattan` as the method. The output of `dist` should be used as the input of `hclust`, and the output of `hclust` should be used as input for `plclust` to produce the dendrogram. Does the addition of *Pongo* change the relationships among the other OTUs (Fig. 10.5)?

Exercise 9. Perform hierarchical clustering using the scaled data for *H. habilis*, *A. robustus*, *A. boisei*, *A. africanus*, *A. afarensis*, and *P. troglodytes* presented in Section 10.5. Use the euclidean distance metric and complete linkage clustering. Repeat the clustering six times more, leaving out a different OTU each time. Are the relationships implied by clustering using all of the data robust?

Exercise 10. As implemented by R, *K*-means with `centers` = number of clusters randomly selects rows of the data matrix as initial cluster centers. Test the robustness of *K*-means in R for the scaled hominoid data in Section 10.5 by repeatedly performing the clustering while specifying 2 for `centers`. What is there about this data set that produces two different pairs of clusters for $k = 2$? Which result is “better”? What do you conclude about the robustness of *K*-means with this method of specifying initial cluster centers?

Exercise 11. Suppose that a set of OTUs measured on n characters have been clustered by *K*-means into three clusters A, B, and C, with centroids defined by $\{a_i\}$, $\{b_i\}$, and $\{c_i\}$, respectively. Values for within-cluster sums of squares for A, B, and C are s_A^2 , s_B^2 , and s_C^2 , respectively.

- a. Write the statistic(s) that should be calculated for classifying OTU X into one of the three existing clusters.
- b. What statistic(s) could be used to determine whether X should be placed separately into a cluster different from A, B, or C?

Measuring Expression of Genome Information

11.1 The Biological Problem

Many biological questions are framed in terms of biochemistry and genetics. Multicellular eukaryotes have between 10^4 and 10^5 genes, many of which code for enzymes and other proteins involved in biochemical processes and their control. DNA in somatic cells is the same regardless of tissue type or physiological state. (Examples of human cells that *differ* in DNA content are anucleate erythrocytes and gametes lacking either X or Y chromosomes.) Even though all genes are generally found in all cells, not all genes are expressed at any one time, nor are all genes expressed in every cell. For example, hemoglobin is not produced in epithelial (skin) cells, and estradiol is not produced in brain glial cells.

To understand the functions of cells, tissues, and organs in complex organisms, it is necessary to know what genes are expressed under different conditions. During the **cell cycle** (the cellular processes involved in duplication of chromosomes and cell division), particular sets of genes are turned on or turned off in a controlled chronological sequence. As we already indicated, genes may be **differentially expressed** (i.e., have different expression patterns) in different tissues in the adult plant or animal. A special case of differential expression occurs during development, when genes are turned on and off in an exquisitely choreographed pattern during the stepwise production of various embryonic stages. At the other end of the spectrum, gene expression patterns in cancer cells differ from those found in normal cells. Alterations in expression patterns of normal or cancer cells treated with a pharmaceutical agent may be a useful guide for assessing that agent's toxicity or efficacy prior to initiation of clinical trials.

How can we measure gene expression? One way is simply to analyze what proteins are present at any particular time. The collection of proteins present in a particular cell type under a particular set of physiological conditions is called the **proteome**. Presently, proteome analysis is not easy to perform in a highly parallel fashion because of limitations in the technology for identifying

and quantifying each protein. Another way to analyze gene expression is by enumerating the abundances of mRNA species in cells. The collection of RNA species present in a cell type for a particular physiological state is called the **transcriptome**. This is much easier to measure in a highly parallel, high-throughput manner, and such measurements are widely employed in genomic, clinical, and drug discovery settings. The basic question answered in transcriptome analysis is, “Is the transcript for gene i in cell type A for condition X more or less abundant than in cell type B for condition Y.” A and B might be normal and cancer cells, respectively, with conditions X and Y identical. Or A and B might both be cancer cells, with condition X representing treatment with 5-fluorouracil and condition Y representing a control.

Transcriptome analysis is widely employed because measurements can be made using parallel and automatable approaches. Moreover, it is well-known that significant gene expression control is exerted at the transcriptional level and at the level of splicing and mRNA processing. But this is not the whole story: production of the actual gene products can also be regulated at the *translational* level and at the level of *protein modification and degradation*. Measuring gene expression from assays of transcript abundance is analogous to measuring the productivity of a law office based upon the number of reams of paper put through the office copier. There is clearly a relationship between the numbers of wills and legal briefs produced and the number of copies made, but it is not a perfect correlation.

Estimates of how well protein levels correlate with mRNA levels differ, partly because of differences in experimental approaches. As shown in Fig. 11.1, there is excellent correlation (correlation coefficient $r = 0.9$) between protein and transcript amounts for abundant transcripts in yeast cells. In contrast, protein levels of low-abundance transcripts do not show a high level of correlation with mRNA levels (correlation coefficient $r = 0.2$; Gygi et al., 1999). In contrast, Futcher et al. (1999) obtained $r = 0.76$ for log-transformed protein and mRNA abundance data, with no significant difference in contributions of low- and high-abundance species. For 80 *E. coli* genes, the comparison between mRNA and protein levels yielded $r = 0.67$ (Arfin et al., 2000). A comprehensive study of the yeast proteome using sensitive antibody detection methods (Ghaemmaghmi et al., 2003) reported a Spearman rank correlation coefficient of $r = 0.57$ between levels of protein and their corresponding mRNA levels. We point these facts out not as a criticism of measuring transcript levels but only as a reminder that this is exactly what is measured: transcript levels—not gene product levels.

11.2 How Are Transcript Levels Measured?

Gene transcript analysis can be performed by using *open architecture* or *closed architecture* approaches. Open architecture methods are independent of prior experimental data (e.g., genome sequence or EST data; see Section 1.5 for

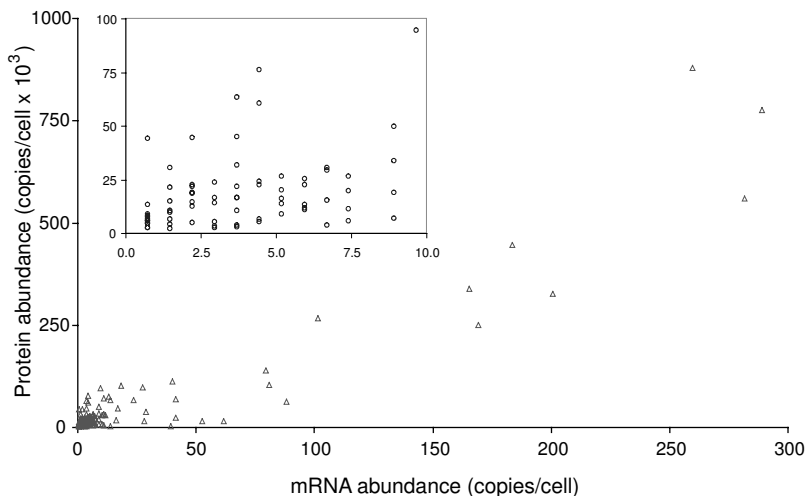


Fig. 11.1. Relationship between protein and mRNA abundances. The amounts of protein per cell for abundant transcripts (more than 75 copies per cell) show a high degree of correlation with the amount of corresponding mRNA. Correlation between protein amounts and mRNA levels is much poorer for low-abundance transcripts (inset box). Reprinted, with permission, from Gygi SP et al. (1999) *Molecular and Cellular Biology* 19:1720–1730. Copyright © 1999 American Society for Microbiology.

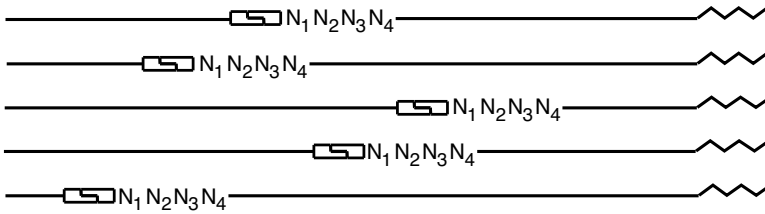
a definition of an EST). In contrast, closed architecture approaches focus on a predetermined set of categorical data (e.g., annotated ESTs). SAGE and TOGA are examples of open architecture approaches. Both apply particular cloning and amplification procedures for sampling a portion of each mRNA sequence. Spotted microarray and oligonucleotide chip technologies are closed architecture systems in the sense that the probes are determined from prior knowledge, as represented in sequence and EST databases, for example. In the case of EST data, *someone else* did the cloning of the cDNA sequences. Although we concentrate on the closed systems because they are currently more extensively employed, we provide a brief description of SAGE and TOGA here.

TOGA (TOtal Gene expression Analysis) (Sutcliffe et al., 2000) associates with each detectable mRNA species a *digital sequence tag*, which depends upon a particular set of octamer sequences and their distances from the 3' end of the mRNA. The starting material is polyadenylated mRNA extracted from any eukaryotic organism. After an initial reverse transcription to produce cDNA (including a *NotI* sequence in the primer supplied), digestion with an enzyme such as *MspI* (recognition sequence 5'-C⁺CGG-3') produces a fragment of characteristic length from the portion of each cDNA adjacent to the 3' end of the mRNA (Fig. 11.2). These fragments are cloned and subjected

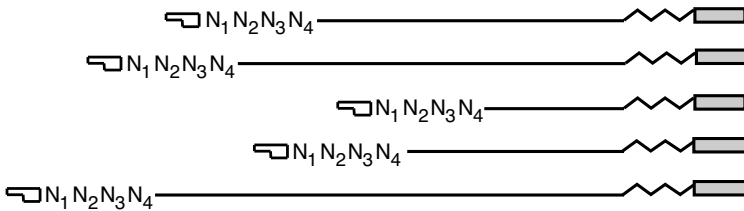
to further amplification steps, including two PCR reactions. The key is that while the PCR steps employ a fixed 3' primer, 5' primers are chosen from one of 256 possibilities that include the position of the *MspI* site and the next four nucleotides $N_1N_2N_3N_4$. That means that 256 (4^4) independent PCR reactions are performed. Each of these PCR reactions contains pools of product that share the same octamer sequence tag at the 5' end (5'-CCGGN₁N₂N₃N₄-3') but differ in the distance of this tag from the 3' end (i.e., each $N_1N_2N_3N_4$ probes a different *MspI* site, producing a PCR product whose length differs from the others). The products within each pool are resolved by gel electrophoresis, and the fragment lengths and amounts within each reaction sample are measured. There are typically 20 to 40 peaks resolved per sample, and it is this step that yields parallel information (sizes) on the products in the product pool. Endonucleases other than *MspI* are used to detect those cDNAs for which the *MspI* site closest to the 3' end is too distant for unambiguous electrophoretic detection or size analysis. Note that this method does not produce the entire sequence of the mRNA and that it uses different PCR primers to detect products having different octamer sequence tags.

SAGE (Serial Analysis of Gene Expression) (Velculescu et al., 1995) is a more widely used open approach. Like TOGA, it derives local sequence information from near the 3' end of the mRNA, but instead of using PCR for sequence discrimination, SAGE employs DNA sequencing. With SAGE, cDNA corresponding to a polyadenylated mRNA mixture is cleaved with a restriction endonuclease that recognizes 4 bp (*MspI*, recognition sequence 5'-C[↓]CGG-3' or *NlaI*, recognition sequence 5'-GATC[↓]-3', for example). Such sites occur, on average, once within the first 256 bp of cDNA upstream of the 3' end. These enzymes are called *anchoring enzymes*. After purification of the

Fig. 11.2 (opposite page). Principles of TOGA. A set of polyadenylated (zig-zag lines) transcripts are represented at the top. They contain sites for cleavage by a tagging restriction endonuclease (or tagging enzyme: box with offset line; only the site closest to the 3' end is indicated). The four nucleotides $N_1N_2N_3N_4$ immediately to the 3' (right) side of the tagging enzyme site may vary from molecule to molecule. Reverse transcription (step 1) using a primer with an added sequence to be used as a right-end primer in a later step (grey box) yields cDNA versions of the transcripts. These cDNAs are digested with the tagging enzyme and *NotI* (step 2). *NotI* cuts the right-end primer sequence, but only very rarely in the interval between the tagging enzyme site and the ends of the cDNAs corresponding to the 3' ends of the mRNAs. The digestion products are cloned and amplified (step 3) and then divided into 256 different aliquots, each of which is used as the substrate for a PCR reaction. Each PCR reaction employs the same right-end primer but a distinctive left-end primer ending in one of the 256 different nucleotide combinations represented by $N_1N_2N_3N_4$. The resulting product molecules are thus distinguished by the tagging enzyme site + $N_1N_2N_3N_4$ and by the distance from the tagging enzyme site to the cDNA end corresponding to the 3' end of the mRNA. Product sizes are determined electrophoretically.

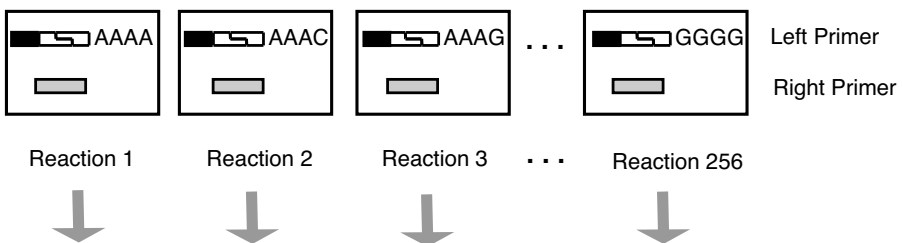


1. Reverse transcribe with primer containing Not I site.
2. Digest with tagging enzyme + Not I.



3. Cloning + multiple amplification steps

4. Divide into 256 aliquots. Perform 256 independent PCR reactions.



5. Analyze fragment sizes from each reaction (256 independent electrophoresis experiments).

cleaved 3' ends, the sample is divided into two pools, A and B, each of which is ligated to an adapter/linker that restores the anchoring enzyme site and also supplies the recognition site for a Type IIS restriction endonuclease, such as *FokI* or *BsmFI* (Fig. 11.3). These enzymes are called *tagging enzymes*. The Type IIS endonucleases do not cut within their recognition sequences but instead cut to the right (in the 3' direction relative to the top strand). *BsmFI*, for example, cuts 10 positions to the right on the top strand and 14 positions to the right on the bottom strand. The key point is that the cleavage is not within the recognition sequence but within an adjacent sequence and occurs a short distance away (10–14 positions), leaving a single-strand overhang. After filling in the end by in vitro DNA synthesis, the product contains (left to right) the adapter A (or B), the tagging enzyme site, the anchoring enzyme site, and a short specific sequence from the cDNA, terminating in a blunt end. The products generated from the A pool are ligated with products of the B pool (at their blunt ends), so that the specific sequences are joined tail-to-tail. PCR primers corresponding to adapters A and B can be used to amplify the ligated tail-to-tail products, and digestion with the anchoring enzyme “pops out” a segment called a *ditag* because it usually contains specific sequences from two different mRNAs. The ditags are concatenated by ligation and cloned into small plasmid vectors to yield product inserts consisting of 15 to 25 ditags. Each ditag has the same length, and it is separated from its neighbors by the site for the anchoring enzyme. The inserts are sequenced, producing a serial readout of sequence tags. The 14-word from each half of the ditag corresponds to a particular mRNA species. The abundance of each mRNA is proportional to the number of occurrences of its 14-word tag. At 50% G+C, the probability of encountering each 14-word starting at any position in a string is $(1/4)^{14}$, or about 1 in 2.7×10^8 . Since we are looking at sequences on average 256 bp from the end of the 3' end of the gene and there are about 3×10^4 genes in a vertebrate genome, the totality of genes subjected to this process would

Fig. 11.3 (opposite page). Principles of SAGE (Serial Analysis of Gene Expression). Two cDNA species are shown (black and grey lines) with an associated extension corresponding to a polyA addition at the 3' end (zig-zag line). The anchoring enzyme (box with offset in middle) usually cleaves the cDNA in several locations, but only the one closest to the 3' end is shown. The two linkers with distinguishing sequences at the 5' end (vertical and diagonal hatching) also contain a portion of the anchoring enzyme sequence and a sequence corresponding to the tagging enzyme recognition sequence (black box). Cleavage with the tagging enzyme (downward arrows) occurs to the right of the recognition sequence and generates fragments that are treated to form blunt ends. These fragments are ligated tail to tail (steps 4 and 5). The product (containing the ditag) can be amplified by using primers corresponding to the linker-specific sequences. Digestion with the anchoring enzyme releases the ditag, which is ligated to other ditags from other cDNA molecules in the sample to form a concatamer that can be cloned and sequenced.



1. Digest with anchoring enzyme.
2. Ligate linkers.



3. Digest with tagging enzyme.



4. Fill in ends.
5. Ligate and amplify.



6. Digest with anchoring enzyme.



7. Ligate ditags to form concatamers.



contribute only about 7.7×10^6 bp of coding sequence. It is evident that there is a comparatively low probability that two or more messages share the same 14-word tag this close to the 3' end of the mRNA. In other words, there is a nearly one-to-one correspondence between a given 14-word and a single gene.

For both TOGA and SAGE, the sequences of the mRNAs are unknown both at the beginning and at the end; only a small part of the mRNA sequence is actually determined. Of course, SAGE and TOGA results can be associated with database entries, but the distinctive feature of these methods is that they operate without prior sequence information. Both of these methods employ cloning, amplification, and other biochemical procedures “up front” to create “subsequence libraries,” and of course they are subject to misincorporation errors during the PCR steps. For closed architecture approaches, these sorts of activities had to be undertaken (by somebody else) to supply all of the EST database entries, but they are invisible to the investigator downloading sequences from curated databases. SAGE is experimentally accessible to any laboratory that has facilities for DNA sequencing and for constructing and handling of small plasmid libraries.

11.3 Principles and Practice of Microarray Analysis

Spotted or oligonucleotide microarrays are by far the most common tools currently used for gene expression studies. These employ EST and genomic sequence data (i.e., they rely on other bodies of experimental knowledge), and they can be performed in a massively parallel fashion. (Over $10^4 - 10^5$ features on an array can be examined simultaneously.) These features are displayed in an area whose size is approximately $1-2 \text{ cm}^2$. Production of microarrays has become highly automated. Optical detection methods are usually (but not always) employed, and optical methods typically have high precisions relative to other methods of data capture. The measured optical signals are readily exported into computer programs for data analysis.

11.3.1 Basics of Nucleic Acids Used for Microarrays

It is important to distinguish between **probes** and **targets**. A probe is a particular DNA sequence corresponding (complementary) to an mRNA whose abundance, presence, or absence within a sample is being evaluated. The actual sequence (especially for oligonucleotide arrays) may be known, but in any event, each probe is characterized by some unique identifying information (e.g., annotation with respect to gene, tissue sample, or clone name). The target is the complex mixture of nucleic acid species being tested for the presence or absence of sequences related to the probe sequence. In most applications, it is the cDNA representation of an mRNA sample isolated from a particular source (e.g., human pancreatic carcinoma cells). In microarray experiments, the probes are immobilized in a grid of positions on a substrate (usually glass

or quartz, but sometimes nylon filters). Each gridded probe sample is a **feature**, which is indexed by its position within the array. The probe sequences for spotted microarrays are DNA molecules (> 200 nt in length) that may be taken from cDNA clones (ESTs), particularly when eukaryotic gene expression is being studied. PCR may be used to create *gene-specific probes* for gene regions that lack introns (e.g., prokaryotic genes, most yeast genes, or exons). Prior to hybridization, the duplex probe and target DNAs are denatured to produce single-stranded molecules suitable for hybridization. For oligonucleotide microarrays, probes of about 25–60 bases are synthesized in situ by combinatorial photolithography to produce gene “chips.” (GeneChip® is a registered trademark of Affymetrix, Inc.)

Eukaryotic mRNAs, which have poly-A extensions on the 3' end, can be isolated by using oligo-d(T) columns, and cDNA can be produced from purified mRNA by reverse transcription (Fig 1.11). The target is a complex mixture of species (perhaps 10^4 different species or more) that depends upon the organism, the tissue, and the physiological condition of the tissue at the time that the RNA is extracted. Different mRNA species may have very different abundances. For example, mRNA for housekeeping genes may be present in many copies, while mRNA for developmental genes may be very low in adult organisms. Typical methods allow detection of species present at a level of one transcript per cell in 10^6 cells (or about 20 pg of transcript per 20 μg of RNA sample). The target molecules may be radioactively labeled, but now it is far more common for them to be labeled with fluorescent dyes Cy3 or Cy5.

The specific interaction between probe and target species is based upon DNA **hybridization**. Given two single-strand DNA molecules X and Y, it is possible for them to hybridize or form a duplex if the sequence of X contains a string of bases that is complementary to all or part of the sequence of Y. By complementary we mean capable of base pairing in the Watson-Crick fashion, with the two strands antiparallel. This process is called hybridization because the two strands need not come from the original duplex molecule. Hybridization is the reverse of denaturation, which for typical DNA under ordinary salt conditions, pH 7.0, occurs in the range of 80°C – 100°C , depending upon base composition (lower for lower %G+C). The temperature at which DNA “melts” (i.e., duplex goes to single-strand) is called T_m . Hybridization reactions are typically fastest at about $T_m - 25^\circ\text{C}$. Hybridizations can be performed at lower temperatures by adding denaturants such as formamide to the hybridization solution. When labeled target molecules hybridize to a particular feature, the fluorescent label on the target species makes the feature capable of fluorescence when it is excited by light of an appropriate wavelength. The amplitude of the signal is proportional to the amount of hybridized target species. If the species is rare, the signal is correspondingly faint.

11.3.2 Making and Using Spotted Microarrays

Much of what has been said up to now applies to both oligonucleotide and spotted microarrays. Both standardized and custom-designed oligonucleotide arrays can be produced commercially. However, many investigators prefer the spotted microarray format because of their ability to customize the arrays easily and to reduce expense. We focus on spotted microarrays. There is a considerable body of literature describing how to deal with various sources of experimental variation in spotted microarray experiments. The general experimental design is indicated in Fig. 11.4.

The probes are typically spotted onto a treated $1'' \times 3''$ glass slide using a cluster of pins that dip into the wells of a microtitre plate (96 wells are common, but 384- or 1536-well plates are also available). The wells in the microtitre plate contain the probe solutions. For 20,000 features, it is necessary to print from about 200 96-well microtitre trays, so robotic printing is usually employed. The other reason for using robotics is that very precise printing (e.g., ± 2 micron error in spot placement) is required to obtain a high density of features in a small area. Another method for emplacement of features employs ink-jet technology, but currently this method is primarily available to firms providing custom-manufactured arrays and not to individual investigators.

Often we wish to understand how gene expression differs for two different conditions (malignant versus normal cells, for example). In these cases, mRNAs from the two different conditions are extracted, differentially labeled, and hybridized together to the microarray. Labeling is usually with fluorescent dyes Cy3 or Cy5, which may be covalently attached to dUTP and employed as one of the substrates during reverse transcription. Alternatively, reactive groups may be attached to the dUTP, and the required dye can later be covalently attached to molecules in each sample prior to mixing. Hybridizing the samples simultaneously ensures that the hybridization conditions are identical, although, as we shall see later, the dyes may affect the hybridization efficiencies differently.

The hybridized array is scanned by a slide reader that illuminates the hybridized spots, stimulating fluorescence characteristic of Cy3 or Cy5. The Cy3 emission maximum is at about 570 nm ("green"), and the Cy5 maximum is near 670 nm ("red"). The amount of fluorescence intensity corresponding to each condition at each microarray feature can therefore be detected. If cDNA derived from RNA expressed during condition X is labeled with Cy3 and cDNA derived from condition Y is labeled with Cy5, an excess of green over red fluorescence at any feature indicates that the gene corresponding to that spot is more highly expressed under condition X than under condition Y. The intensities of light in appropriate wavelength ranges are measured for pixels over the entire slide and stored in the computer. They are, in effect, an image of the detected fluorescence intensities. For each spot (feature), it is necessary to decide over which area of the slide to count the fluorescence intensity. If the feature has an irregular shape or aberrant size, we may be

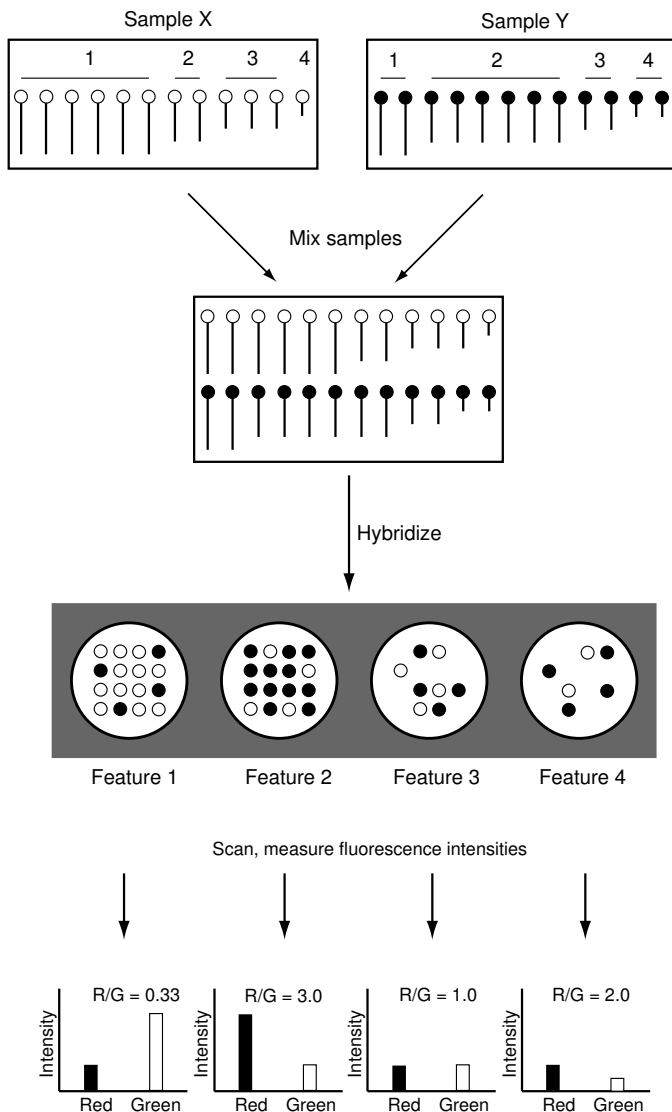


Fig. 11.4. Principles of spotted microarrays. Samples X and Y are cDNA representations of transcript samples isolated under two different conditions. The same four particular species are shown for each sample, and their abundances may or may not differ under the two conditions. Sample X has been labeled with Cy3 (open circles: produces green fluorescence), and sample Y has been labeled with Cy5 (filled circles: produces red fluorescence). The mixture of equal amounts of X and Y is hybridized to a spotted microarray (four features are indicated). The number of open or filled circles in each feature represents the amount of Cy3- or Cy5-labeled target species hybridized. The fluorescence intensities observed for each feature in each of the two color channels are indicated at the bottom of the illustration.

counting intensity from a portion of the slide where no cDNA was spotted, leading to a low reading. Also, there may be residual nonspecific binding of hybridization solution not attributable to a specific probe. This produces a fluorescent background intensity, which should be subtracted from the spot intensities. For this reason, there is automated software for locating the spot, delineating its boundaries, and measuring the background to be subtracted prior to recording the intensity at each wavelength.

An example of the application of this technology is shown in Fig. 11.5 (Arbeitman et al., 2004). In this example, mRNA samples from *dsd^D* mutant *D. melanogaster* (fruit flies) (labeled with Cy5) and from normal adults (labeled with Cy3) were each reverse transcribed to cDNA and labeled with the indicated fluorophore prior to hybridization. Panel A shows one of 12 blocks of 24×24 features from this particular experiment. Red spots correspond to genes whose expression is higher in the mutant, and green spots correspond to genes with reduced expression in the mutant. Panel B illustrates some artifacts that can appear in such experiments. Because such artifacts can occur, microarray images must be inspected by humans before data are processed. (Complete automation is not currently feasible.)

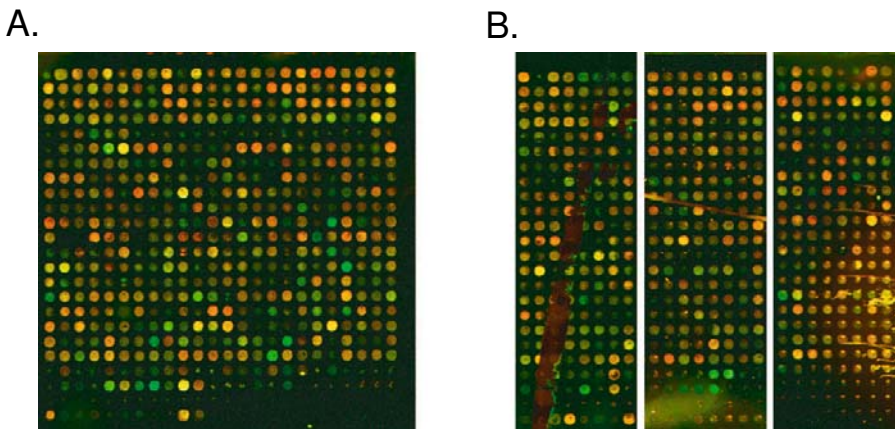


Fig. 11.5. [This figure also appears in the color insert.] Spotted microarray hybridized with target sample representing *Drosophila melanogaster* transcripts. Red spots correspond to genes whose expression is increased and green spots to genes whose expression is decreased in comparison with the control. Yellow spots indicate no change. Panel A: Normal 24×24 block of features. Panel B: Portions of blocks displaying various artifacts, including a tear in the polylysine coating the glass slide (vertical jagged brown patch, left section), a possible dust particle (linear feature, center section), and a possible evaporation artifact from a block near the edge of the array (yellowish patch with streaks, right section). The greenish fluorescence at the bottom of the center section came from contaminating material in DNA samples prepared by nonstandard methods. (Image provided by Dr. Michelle Arbeitman, University of Southern California.)

11.4 Analysis of Microarray Data

There are two very different computational issues associated with microarray analysis. The first is processing of the experimental data to produce a result that accurately reflects either absolute amounts of transcripts in cells or, more commonly, the ratios of these amounts under two different experimental conditions. The result of processing the data is a **gene expression matrix**, one form of which is composed of n rows, each corresponding to a gene or feature on a microarray, and m columns, each of which corresponds to a condition (e.g., a time point) for which expression levels were measured. The content of each element of the gene expression matrix is either a fluorescence intensity or a ratio of two fluorescence intensities. The second computational problem is interpretation of the intensity data in the gene expression matrix to provide biological insight or as a guide to the design of additional experiments. We treat the processing of the experimental data in this section and discuss data interpretation in Section 11.5.

11.4.1 Normalization

Experimental intensity data ordinarily require processing before they are converted to expression matrix entries. The need for this processing is revealed by control experiments. Consider a control experiment for which a single mRNA sample is divided into two equivalent portions. One of these is labeled with Cy3 and the other with Cy5. These two labeled halves of the original sample are then mixed and hybridized competitively to probes on a microarray. The concentration of species that can hybridize to any particular probe spot is identical for each of the two mRNA samples by experimental design. Therefore, we would expect that the fluorescent intensities detected in the Cy3 and Cy5 channels would be identical. They are not: there usually is a **dye bias** that needs to be corrected. Dye bias can result from differences in the incorporation efficiencies for Cy3-dUTP and Cy5-dUTP during reverse transcription, from differences in hybridization efficiencies for molecules having different dye labels, and from differences in the fluorescent properties of the two dyes.

Adjusting the data to remove biases from dye labeling or other experimental parameters is called **normalization**. In the perfect control experiment (equal concentrations of each species and no dye bias or other systematic variation), we would expect a plot of Cy5 intensity versus Cy3 intensity for spots in a microarray experiment to fall on a straight line having a slope of 1.0. In fact, Cy5 intensities are systematically lower than Cy3 intensities when equivalent amounts of sample are present. If we indicate Cy5 intensities by R (“red”) and Cy3 intensities by G (“green”), a regression of R against G normally produces a slope k that is less than 1.0. If the regression is linear, then usually

$$R = kG, k < 1.0.$$

To correct the observed R values so that intensities represent the abundance of mRNA species in the sample, we should multiply R by $1/k$:

$$R_{\text{corr}} = k^{-1}R$$

or

$$\log_2(R/G) = \log_2(kG/G) = \log_2 k.$$

In a “real” experiment, for which mRNA samples were extracted from cells under two different biological conditions, k can be reasonably estimated by linear regression of R against G for *all* features. This approximation is called **global normalization**. It is valid because *most* genes are not differentially expressed, and among the small number of genes that are differentially expressed, roughly the same number of genes will be up-regulated as down-regulated.

But there is an additional complication: even if there is no change in expression level for a set of features, the dye bias in $\log_2(R/G)$ is not constant but varies as a function of intensity. This type of systematic variation is revealed by “MA plots” (Yang et al., 2002). M is defined as $\log_2(R/G)$ (i.e., $\log_2 R - \log_2 G$), and $A = (\log_2 R + \log_2 G)/2 = \log_2(RG)/2$. A is the logarithm (base 2) of the geometric mean of the intensities, corresponding to an average of the logarithms of the intensities. An example of an MA plot is shown in Fig. 11.6A, and a plot of the same data after correction is shown in Fig. 11.6C. When all such appropriate corrections have been made, the average value of M is 0.0 for all values of A . This is called **intensity-dependent normalization**. More complicated normalization procedures take into account the particular pin used to print particular sets of spots and other sources of variation (Yang et al., 2002). Computational Example 11.1 illustrates global and intensity-dependent normalization using real data.

Computational Example 11.1: Global and intensity-dependent normalization of microarray data

Step 1: Examining the data

The data we use were collected for the dsx^D mutant *Drosophila* described above. The array shown in Fig. 11.5A represents one of 11 usable blocks from this experiment. The data from the scanner appear in a spreadsheet containing 9216 rows of entries (6912 rows with nonzero entries for intensities) and 43 columns, plus header information. In addition to header information describing overall conditions of the experiment, the data matrix consists of columns describing block numbers, columns and rows of spots within blocks, spot positions, various data related to the means and medians of spot intensities, and background intensities at the two wavelengths. Rows correspond to different genes or control spots, and rows that are unreliable or that have zero intensity were “flagged” by the scanning software. The complete data set can be downloaded from <http://www.cmb.usc.edu>. We use only the median

intensities measured at two wavelengths, corrected for background. The data are preprocessed as described in Appendix C.5 to produce a matrix (5640×6) containing A and M values for plotting. This matrix can also be downloaded as a text file from the URL indicated. Only unflagged rows are used, and the columns are, respectively, the spot identity (ID), red intensity, green intensity, flag, A , and M . The first three rows of data are shown below.

```
# array.a.m is the data matrix
> array.a.m[1:3,]
      V1    V2    V3 V4      a      m
1 GH01040 19404 6040  0 13.402200  1.6837336
2 GH01059 12628 3352  0 12.667572  1.9135321
3 GH01066  2236  893  0 10.464610  1.3241881
```

Step 2: Global normalization factor

We showed above that $\log_2(R/G)$ is roughly equal to $\log_2 k$. Therefore, $k = 2^M$. We use the average value of M to determine the normalization factor:

```
> mean(array.a.m[,6])
[1] 0.2903073
> 2^mean(array.a.m[,6])
[1] 1.222901
```

This means that the R intensity is, on average, 22.3% greater than the G intensity for all features, most of which are expected not to differ. For global normalization, corrected R intensity values should be obtained by multiplying all R values by $1/1.2229$, or 0.8177 . It is not necessary to use this normalization if the intensity-dependent procedure described below is used.

Step 3: Intensity-dependent normalization

We take a quick look at the data as an MA plot:

```
#Set parameters for plotting three graphs on the same sheet
>par(pin=c(4,2),mfrow=c(3,1))
#Initial plot to look at the data
>plot(array.a.m[,5],array.a.m[,6],pch=".",xlab="A",ylab="M")
```

The plot (Fig. 11.5A) shows an upward trend in M values for larger values of A . This systematic error needs to be corrected. We do this by calculating an “average” curve through the data and subtracting the predicted value at each A from the actual value. There are two functions in the basic R package that can be used: `lowess()` and `loess()`. The former stands for “locally weighted scatterplot smoother,” and the latter is a different smoother that locally fits scatterplots to polynomials. We won’t go into detail here, but you should check the R documentation for a description of these methods and default parameter settings.

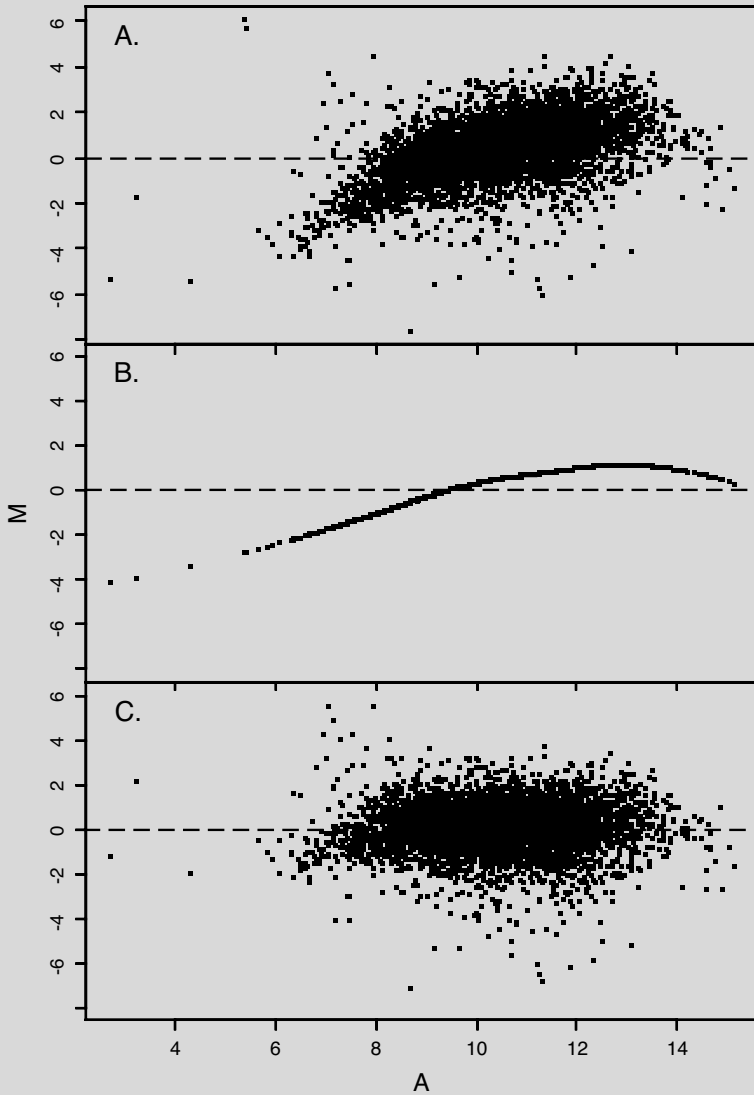


Fig. 11.6 MA plots showing the effect of intensity-dependent normalization. Panel A: Data prior to normalization. Panel B: Predicted `loess` curve based on data in panel A. Panel C: Data after subtraction of predicted `loess` values. Note that the data now cluster around the value $M = 0$, with a slope of zero. The tight clustering of data points around the line $M = 0$ indicates that most genes do not change their expression levels much under the two conditions compared (dsx^D pseudomales versus dsx null females). (Data supplied by Dr. M. Arbeitman, University of Southern California.)

Applying the `loess()` function:

```
> MA.ls<-loess(array.a.m[,6]~array.a.m[,5])
#Note argument order: dependent variable is listed first.

> tmp<-predict(MA.ls,array.a.m[,5])
# tmp is a vector of predicted values
# predict is a standard R function
> length(tmp)
[1] 5640
#Checking that correct number of data are returned.
```

Plot the predicted loess curve (Fig. 12.5B):

```
> plot(array.a.m[,5],tmp,pch=".",xlab="A",ylab="M",
+ ylim=c(-8,6)) #Plot with same scale used in first panel.
```

Apply the correction to the data by subtracting the loess-predicted value from every M datum:

```
> MA.norm<-array.a.m
> MA.norm[,6]<-MA.norm[,6]-tmp #Subtracting predicted value
> mean(MA.norm[,6])
[1] -0.003620717
#Check: This should be close to zero after correction.
```

Plot the corrected data (Fig. 11.6C):

```
> plot(MA.norm[,5],MA.norm[,6],pch=".",xlab="A",ylab="M",
+ ylim=c(-8,6))
```

Note that the data are now clustered about a line with slope zero and y intercept zero. Furthermore, note that we did not use globally normalized data. The procedure automatically places the mean value of all $\log_2(R/G)_{\text{norm}}$ close to zero.

11.4.2 Statistical Background

In Chapters 2 and 3, we introduced the concepts of mean and variance for samples drawn from a large population. We also discussed the Central Limit Theorem, which indicated that as the sample size increases, the probability distribution for sample means approaches the normal distribution. When considering microarray data, there is an additional complication that must be considered: we probably will not know the standard deviation for the measurement of intensity or intensity ratios for each spot. However, with appropriate replicate experiments, it is possible to estimate the sample standard deviation.

We should perform a number of replicates for each microarray experiment, but because of expense or limited samples, this number may be low. As a

consequence, the estimates of average expression ratio and the standard error in the expression ratio may not be reliable. The formalism that we discussed previously employed the population mean and standard deviation. Now we must manage with the average of a small number of measurements for each spot instead of a mean and with the sample standard deviation instead of a population standard deviation. Recall that the estimate of the population variance based upon a sample is given by

$$s^2 = \frac{1}{n-1} \sum_{i=1}^n (X_i - \bar{X})^2, \quad (11.1)$$

where \bar{X} is the sample mean and n is the sample size. The quantity s is called the sample standard deviation. For microarray experiments, the number of replicates, n , will often be a small number much less than 10. Although there may be on the order of 10^4 spots on the microarray, the variance for each spot generally depends upon its mean expression level (Long et al., 2001), so it may be inappropriate to estimate s^2 for any particular feature by including results from spots having very different expression levels.

It is common to refer to the base or standard condition as the **control** and the condition resulting from experimental or other perturbations as the **treatment**. Suppose that we are interested in the expression level X_j^t of gene j in cancer cells (*treatment*) in comparison with its expression level X_j^c in unaffected cells (*control*). The null hypothesis would be that the expression level is no different in cancer cells compared with unaffected cells, and we would want to perform a hypothesis test to determine whether the observed mean value for X_j^t is significantly different from the mean value for X_j^c . With large numbers of measurements, the sample estimates of the means and standard deviations are good approximations to the population values and the distributions of the means are approximately normal, and therefore the hypothesis could be tested by using the normal distribution in the usual way. When there are few replicates, however, the normal distribution cannot be used, and instead the distribution of the statistic t' is appropriate for testing whether two independent means are different:

$$t' = \frac{\bar{X}_j^t - \bar{X}_j^c}{\sqrt{\frac{s_t^2}{n_t} + \frac{s_c^2}{n_c}}}, \quad (11.2)$$

where n_t and n_c are the numbers of measurements for treatment and control, respectively, and s_c and s_t are the respective sample standard deviations for treatment and control. If the standard deviations of the treatment and control groups are equal, then the difference between the two means can be tested by using the more familiar Student's t statistic,

$$t = \frac{\bar{X}_j^t - \bar{X}_j^c}{s \sqrt{\frac{1}{n_t} + \frac{1}{n_c}}}, \quad (11.3)$$

where s is the standard deviation estimated from the combined samples. The t distribution depends upon the sample size, usually expressed in terms of degrees of freedom. In the expression above, the number of degrees of freedom is

$$\text{df} = n_c + n_t - 2. \quad (11.4)$$

When the sample sizes are identical, (11.3) reduces to

$$t = \frac{\sqrt{n}(\bar{X}_j^t - \bar{X}_j^c)}{s\sqrt{2}}, \quad (11.5)$$

with $\text{df} = 2n - 2$. As the sample size (and number of degrees of freedom) becomes very large, t approaches a normal distribution.

Some investigators claim that if genes display expression intensity ratios that differ by some arbitrary factor (e.g., 2) when two different conditions are compared (e.g., cancer or unaffected), then those genes are differentially expressed. This approximation can only be reliable if the variances are the same for measurements across all genes (an unlikely circumstance) and s is sufficiently small. Since this is not usually the case, more careful hypothesis tests (t tests at a minimum) are required to identify genes whose expressions differ under the two conditions. If the variances are poorly estimated because of insufficient replication of the experiment, then any inferences made from these experiments are correspondingly unreliable. Some Bayesian approaches have been employed for experiments having small numbers of replicates, but this depends upon having a realistic prior probability distribution. The interested reader may consult Long et al. (2001) for an example. However, sophisticated statistical wizardry will not compensate for insufficient replication of experiments.

Microarray experiments involve large numbers of features (spots). This means that there may be a substantial number of features that *appear* to reveal differential expression just by chance. For example, suppose that we have a microarray having 10^4 spots and that we have performed three replicates (and so have $2 \times 3 - 2 = 4$ degrees of freedom). Taking the average value of the intensity ratio of X_j^t to be 2 and the average value of the intensity ratio for X_j^c to be 1 and $s = 0.2$, we ask how many spots would be predicted to differ by a factor of two or more by chance. We calculate that $t = 6.12$. For four degrees of freedom, the probability that $|t| > 6.12$ is around 0.004, which means about 40 spots out of 10,000 will display expression ratios that differ by a factor of two just by chance when the two conditions are compared. These represent false positive errors.

This calculation illustrates the general problem of **multiple hypotheses testing**: even though the probability of obtaining a false positive result for any individual spot may be very small, for a large collection of spots there is a high probability of obtaining a number of false positive results comparable to the expected number of true positive results. A detailed description of testing multiple hypotheses is beyond the scope of this book, but as a starting point

for those who wish to explore this issue further, we note that the **Bonferroni correction** may be employed. The idea is that if the experiment-wide significance level (i.e., the probability of obtaining false-positive results for the experiment as a whole) is chosen to be α_B , then a conservative significance level α for each of the $N = n \times m$ individual features is

$$\alpha = \alpha_B/N.$$

11.4.3 Experimental Design

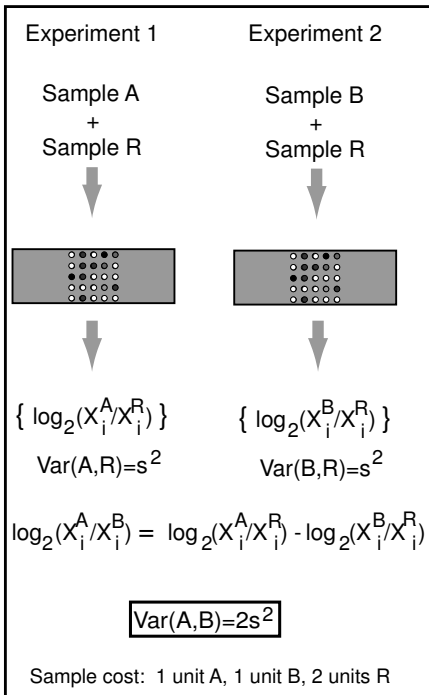
Reliable conclusions based upon microarray data require estimates of the variances for measured intensities or intensity ratios, and the variances depend upon both the experimental design and the number of experimental replicates. It matters *how* the replicate measurements are performed (Churchill, 2002; Yang and Speed, 2002). For microarray experiments, three different types of replication (corresponding to three general sources of variation) can be imagined.

- Replication of samples: Individual biological organisms, even of the same species, strain, age, and sex, are not identical. Data from several individuals drawn from that population are required to make a reliable conclusion about a population.
- Replication of sample preparations (technical replicates): Multiple steps are required to produce the target material for hybridization to an array of probes, and these preparations may be subject to systematic or random errors. We already described dye-specific biases (e.g., identical samples labeled with Cy5 or Cy3 do not produce the same result).
- Replication of slide hybridization and image processing: This takes into account slide-to-slide variation and variation in the printing of individual spots.

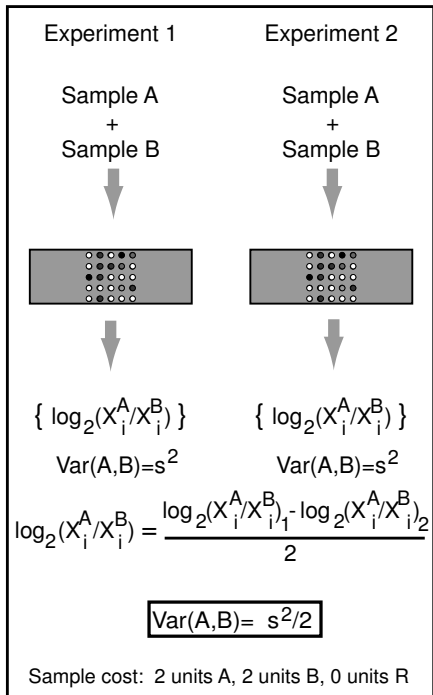
These different sources of variation have very different contributions to the overall variance of any given measurement. For example, replicate probe spots on the same slide yield intensity ratios that have correlation coefficients of about 0.95, replicate probe spots on *different* slides hybridized to the same target sample have correlation coefficients in the range 0.6 to 0.8, and if different sample preparations are used for hybridization, the correlations between intensity ratios for the same spots measured for different samples may

Fig. 11.7 (opposite page). Influence of experimental design on variance in the logarithm of the expression ratio for condition A compared with B ($\log_2(X_i^A/X_i^B)$). Target samples are prepared for conditions A, B, and C or control R. Different experimental designs yield different values of the variance (boxes near the bottom of each panel) and require different amounts of sample. Of these three protocols, protocol B yields the lowest variance but requires twice as much sample.

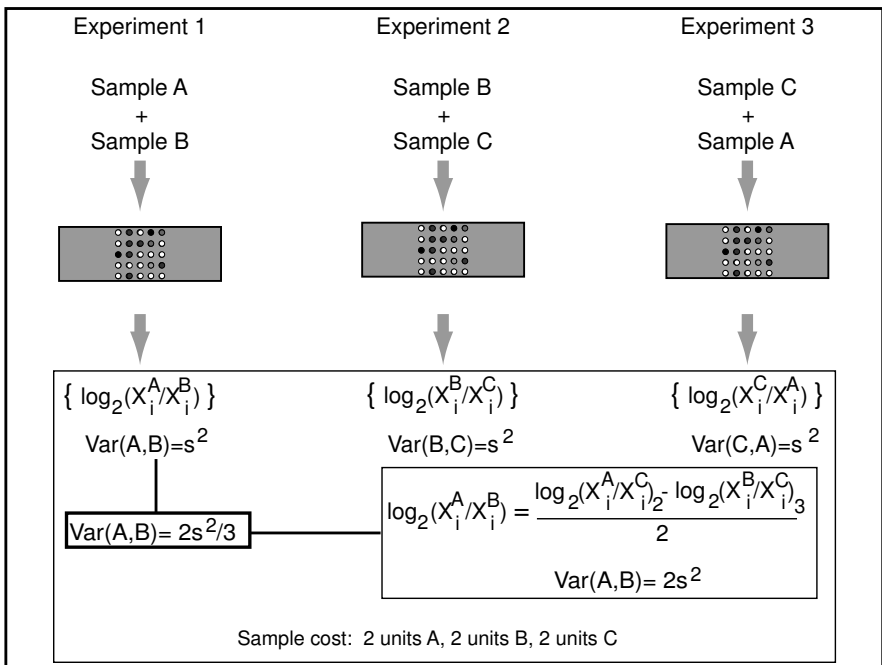
A.



B.



C.



be only 0.3 (Churchill, 2002). We might suppose that oligonucleotide arrays would produce more concordant results because of the thoughtful design and well-defined chemical synthesis of probes. However, direct comparisons of over-expressed genes identified by using the same sample with two different oligonucleotide array platforms produced agreement in only 4–15% ($\alpha = 0.001$) or 14–26% ($\alpha = 0.01$) of the cases (Tan et al., 2003).

Good experimental design principles can help optimize the data that can be extracted by minimizing the variance for the desired experimental quantity (Fig. 11.7). For example, suppose that we are studying target samples A and B and have available reference sample R. Let X_j^A and X_j^B be the intensities measured for gene j under conditions A and B. Denote the variance in $\log_2(X_j^A/X_j^R)$ as s_j^2 . (We are assuming the presence of multiple spots for gene j so that s_j^2 can be estimated.) The log of the intensity ratio can be measured directly as $\log_2(X_j^A/X_j^B)$ if samples A and B are cohybridized in a single experiment, or the ratio can be obtained indirectly if A and B are each hybridized independently of each other but in the presence of reference sample R:

$$\log_2(X_j^A/X_j^B) = \log_2(X_j^A/X_j^R) - \log_2(X_j^B/X_j^R).$$

The indirect approach requires two experiments, and the variance for that case is the sum of the variances of the individual experiments:

$$(s_j^2)_{AB} = (s_j^2)_{AR} + (s_j^2)_{BR}.$$

But if we were going to perform two experiments anyway, we could measure $\log_2(X_j^A/X_j^B)$ directly in two independent hybridizations. The variance (of the average) in that case would be $(s_j^2)_{AB}/2$. The two indicated strategies involve the same investment in labor and material, yet they yield a fourfold difference in the variance of the desired quantity. Some specific examples illustrate how designs can matter.

For example, one common experimental approach is to measure data from samples A, B, and C relative to sample R. R might be a well-characterized biological sample created by pooling a number of smaller samples so that the targets in R all have about the same abundance (yielding the same level of hybridization to all probes). Then we could perform three hybridizations using mixtures A+R, B+R, and C+R, and measure $\log_2(X_i^A/X_i^R)$, $\log_2(X_i^B/X_i^R)$, and $\log_2(X_i^C/X_i^R)$ for each gene. As we indicated above, if we seek relative expression levels in samples A and B, B and C, and A and C, those quantities can be obtained from measurements involving reference sample R by subtraction of the appropriate log terms, with a variance of $2s_i^2$ in each case. The experiments require one sample each of A, B, and C, and three samples of R, for a total of three hybridizations.

There is another way of setting up the experiment. Instead of using R, we could employ the following combinations: A+B, B+C, and A+C. In this case, there are two measurements that produce relative expression levels of

A and B: the A+B experiment, measured directly with a variance of s_i^2 , and the indirect measurement comprised of $\log_2(X_i^A/X_i^C) - \log_2(X_i^B/X_i^C)$, with a variance of $2s_i^2$. The variance for the average of the two measurements (direct and indirect) is

$$s_{\text{avg}}^2 = (s_{\text{direct}}^2/1 + s_{\text{indirect}}^2/2)/3 = s_i^2(1/1 + 2/2)/3 = 0.67s_i^2.$$

Clearly, performing the series of direct measurements without use of a reference produces a result having a lower variance. In this case, however, two samples each of A, B, and C were required, with no samples of R. From the standpoint of reliability of the estimations, elimination of the reference sample is preferable in this case, provided that we have enough sample material. For more complicated experimental designs, see Yang and Speed (2002), from which the examples above were taken.

11.5 Data Interpretation

In the last section, we discussed processing the experimental data to obtain reliable intensities or intensity ratios and their variances to produce gene expression matrices. In this section, we assume that we are given an appropriately corrected gene expression matrix and are asked to analyze the data to provide biological insights. As we shall see, some of the mechanisms for doing this have already been presented in Chapter 10. The approach for data analysis is in part dictated by the purposes of the experiment. Among those purposes are:

- *Annotating* anonymous genes based upon their expression patterns over a number of conditions. For example, if under a variety of conditions gene j shows patterns of expression that are similar to patterns for a set of other genes whose functions are known, then we might hypothesize that j functions in a similar pathway and test this hypothesis experimentally. This is known as “guilt by association.”
- Identifying genes (known or unknown) that are *co-regulated* and that may function in the same biochemical pathway. This may lead to new insights into gene regulation mechanisms and may suggest experiments for analyzing the promoter regions of co-regulated genes for shared collections of transcription factor binding sites.
- *Classifying* biological specimens (e.g., tumors) based upon their gene expression patterns. This could lead to identification of a small number of genetic markers that would be clinically useful for diagnosis.

The analytical approaches fall into one of two categories, distinguished by whether or not information from *outside* the microarray experiment is employed. **Supervised** methods incorporate this prior knowledge by including class labels associated with each feature. For example, we might take genes

of known function (e.g., translation of mRNA, DNA repair) and use their expression patterns (suitably represented) to classify unannotated genes into a defined functional category. As another example, the expression patterns revealed in the mRNA from a collection of tumors might reveal patterns specific to a particular type and stage of tumor. **Unsupervised** methods start with a collection of multivariate data and produce groupings of genes or combinations of variables based upon information inherent in the data without additional outside information. We illustrate unsupervised approaches by clustering methods and principal component analysis (PCA), described below.

The methods for data analysis depend upon the data structure. As we have indicated, the data are contained in a gene expression matrix whose entries correspond to intensities or intensity ratios for each feature (i.e., spot on the microarray). In our discussion so far, we have considered the gene expression matrix to represent data for n genes measured on m conditions, with the data for each gene j entered in row j of the matrix. However, if the purpose of the experiment is tumor diagnosis, it may be of greater interest to consider m rows of conditions (e.g., m rows of tumor samples) measured over n genes (i.e., one column for each gene). The types of variables of greatest concern to the experimenter are called criterion or **response variables**. In Chapter 10, the criterion variables were the OTUs. The variables that are potentially useful for determining the value of the response variable are called **predictor variables** or **attributes** (the characters in Chapter 10). If the purpose of the experiment is classification of genes, then genes are the response variables and the conditions under which their expression was measured are the predictor variables. If the main interest is in classifying conditions or grouping similar conditions, then the conditions are the response variables and the genes are the predictor variables. In other words, either the expression matrix of n genes \times m conditions, or *its transpose*, may be analyzed, depending upon the purpose of the experiment. For each row of the expression matrix (or its transpose), the set of predictor variables for that row is called a **profile**. These profiles can be used for clustering or classification.

We discuss two general approaches to data analysis: clustering and data reduction. In both cases, complex data that are hard for the human mind to comprehend are organized and simplified. For example, arrays containing 10^4 features measured under 50 conditions contain 500,000 entries of numbers measured on a continuous scale. *Clustering* assigns features (genes) into a smaller number of categories, with all members in any category having similar profiles. This allows the profiles of sets of individual genes to be summarized by an average cluster profile. *Data reduction* may allow either elimination of a subset of the predictor variables or perhaps formation of combinations of them that can be used to identify a subset that explains most of the variation in the data.

11.5.1 Clustering of Microarray Expression Data

Clustering methods presented in Chapter 10 (e.g., hierarchical clustering and K -means) can be employed with microarray data to group genes having similar expression patterns into clusters. Eisen et al. (1998) were among the first to use hierarchical clustering to organize microarray data. Recall that with hierarchical clustering, objects are grouped according to their similarities or distances. A number of different measures of similarity (or distance), such as Euclidean distance, could be employed. When comparing expression patterns of two genes i and j measured over m conditions, we can use the Pearson product-moment correlation coefficient as a measure of similarity (cf. Eisen et al., 1998),

$$r_{ij} = \frac{\sum_{k=1}^m (g_{ik} - \bar{g}_i)(g_{jk} - \bar{g}_j)}{(m-1)s_i s_j},$$

where g_{ik} is the expression level for gene i under condition k , \bar{g}_i is the average expression level of gene i , and s_i is the standard deviation over all m conditions. All of the correlation coefficients can be grouped into a correlation matrix, R . (It is evident from Section 10.3.1 in Chapter 10 that there is a relationship between the **correlation coefficient** and the Euclidean distance metric, but we won't go into further detail on this point.) Note that the correlation coefficients are closely related to the **covariance**:

$$\text{Cov}(g_i, g_j) = \frac{\sum_{k=1}^m (g_{ik} - \bar{g}_i)(g_{jk} - \bar{g}_j)}{(m-1)}.$$

The covariances can be grouped together in a **covariance matrix**, S . Note that the diagonal terms of S are the variances of the respective variables. Either the entries in the correlation matrix R or the covariance matrix S could be used for clustering, taking only that portion above or below the diagonal, as we did in Chapter 10. In this case, objects having the greatest similarity (as measured by the correlation coefficients) or, alternatively, the smallest distance would be joined first in hierarchical clustering. The matrix R can be converted into a distance matrix D by changing the signs and adding 1.0 to each of the r_{ij} .

Computational Example 11.2: Clustering of expression data

The data (Appendix C.4) are the mRNA levels for a set of 12 yeast genes measured at 16 successive time points. We use K -means clustering to determine how many different expression patterns there are and to group together genes having similar patterns.

Step 1:

Read the data from a text file into an R matrix from file `yeast_dat`. We use the function `scan` to read the data into a matrix, and then we add the row names.


```

yeast.dat<-matrix(scan("yeast_data"),nrow=12,byrow=T)
clone.name< c("YGR027C","YLR259C","YGL189C","YEL032w",
+ "YPL240C","YLL026w","YLR274W","YBR202w","YER131w",
+ "YDR258c","YBL072c","YBL023c")
dimnames(yeast.dat)<-list(clone.name,NULL)

```

Step 2:

We are interested in the patterns of expression rather than the absolute amounts. Therefore, we standardize the data points for each gene by subtracting from each entry the mean value for that gene and dividing by the standard deviation.

```

> for(i in 1:12){
+ syeast.dat[i,]<-(syeast.dat[i,]-mean(syeast.dat[i,]))/
+ sqrt(var(syeast.dat[i,]))
+ }

```

The first few entries of the standardized matrix `syeast.dat` are shown below.

```

YGR027C  -1.8475332  -0.448629053  0.42444587  -0.1956358  ...
YLR259C   2.6423248   0.553840762  0.47351445  -1.1330117  ...
YGL189C  -0.5141768  -0.994527569  1.25917287  1.5338940  ...
YEL032w  -0.3421999   1.040063427  -0.27039405  -0.5396661  ...
YPL240C   3.3712044   1.212655197  0.25669716  -0.2691817  ...

```

Because of the standardization, the entries for all genes are now all of the same order of magnitude.

Step 3:

Perform K -means clustering for different values of k :

```

# k=2
> ckm2<-kmeans(syeast.dat,2,iter.max=10)
> ckm2
$cluster
 [1] 1 1 1 2 1 1 2 2 1 1 1 2
$centers
      [,1]      [,2]      [,3]      [,4]      [,5]
 1  1.0129085  0.07897436  0.3453561  -0.1251811  0.0009998353...
 2  -0.7827595  0.35531276  -0.0626123  -0.8963423  -1.0668526992...
$withinss
 [1] 82.02579 10.34386
$size
 [1] 8 4

```

As we indicated in Chapter 9, `$` is used as a prefix to indicate each item in a list of objects that are not necessarily of the same type: scalar, vector, matrix, or data frame. `kmeans` produces five quantities as output. `cluster` indicates the cluster to which each row belongs. `centers` contains two 16-dimensional vectors, which are the coordinates of the two cluster centers. `withinss` is the within-cluster sum of squares, a measure of how close the members of each cluster are to the cluster centers. `size` summarizes the number of members in each cluster. In this case, cluster 1 has eight members, and from the `withinss` values it appears that they are on average more diffusely distributed than are the members of cluster 2.

For subsequent iterations, we omit `centers` to conserve space:

```
# k=3
> ckm3<-kmeans(syeast.dat,3,iter.max=10)
> ckm3
$cluster
 [1] 2 1 2 3 1 1 3 3 2 1 2 3

$centers
...
$withinss
 [1] 5.776459 17.329197 10.343858
$size
 [1] 4 4 4
```

Since the `withinss` for the third cluster is the same as for the last cluster with $k = 2$, it is evident that the first cluster obtained with $k = 2$ has been split in two. Notice how the sum of all three `withinss` values has fallen as a consequence of proceeding from $k=2$ to $k=3$.

```
#k=4
> ckm4<-kmeans(syeast.dat,4,iter.max=10)
> ckm4
$cluster
 [1] 4 1 4 2 1 1 3 3 4 1 4 3
$centers
...
$withinss
 [1] 5.776459 0.000000 6.407072 17.329197
$size
 [1] 4 1 3 4
```

By adding a fourth cluster, all that has happened is that one of the members of cluster 2 from $k = 3$ has been defined as a cluster having only one member. (Note that `withinss = 0.0`.)

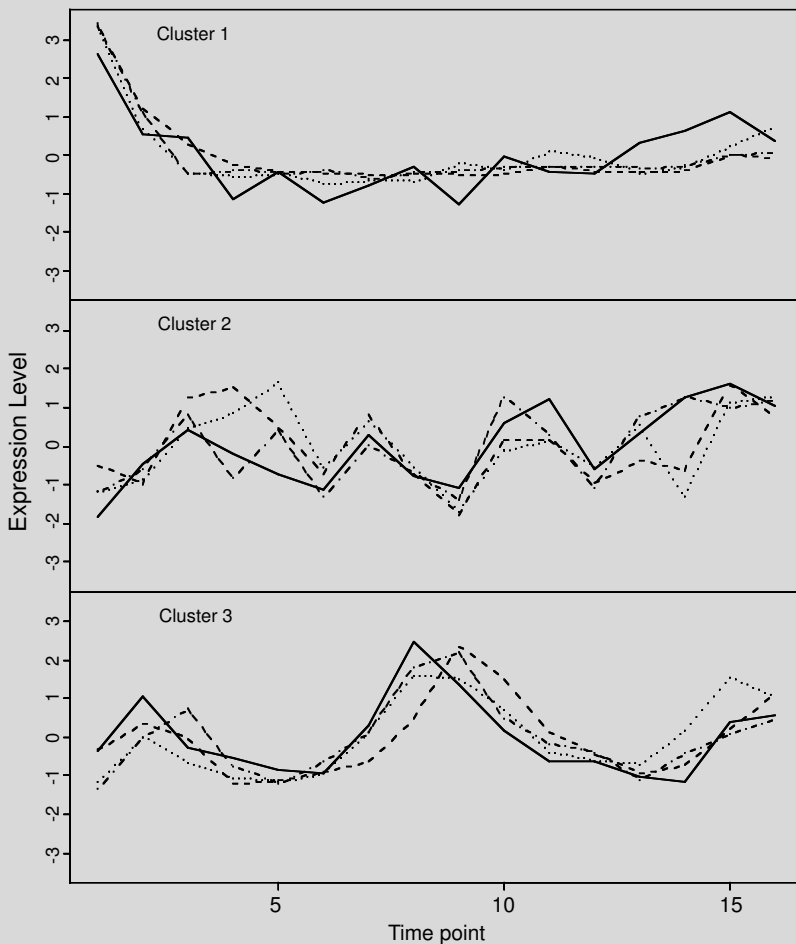


Fig. 11.8. Results of K -means clustering of expression patterns for 12 yeast genes (Computational Example 11.2). The time course of expression is similar for genes within each derived cluster.

If this process is repeated for $k = 5$ and $k = 6$, we can sum the `withinss` values for each value of k and plot them versus k to obtain a plot like Fig. 10.8. We prefer that value of k beyond which the `withinss` does not drop too much for each additional increment in k . In this case, we choose $k = 3$.

Step 4:

Plot the data, $k = 3$, making a separate plot for each cluster (Fig. 11.8). Use `$cluster` identifiers to extract appropriate rows of `syeast.dat`. The three panels were plotted using

```

> par(mfrow=c(3,1))
> plot(syeast.dat[2,],ylim=c(-3.5,3.5),type="l",lty=1,
+ xlab="Time Point", ylab="Expression Level")
> points(syeast.dat[5,],type="l",lty=2)
> points(syeast.dat[6,],type="l",lty=3)
> points(syeast.dat[10,],type="l",lty=4)
> title("Cluster 1")...

```

The lines beginning with `plot` and continuing through `title` are repeated (with appropriate modifications to the arguments) to produce plots of the two other clusters. Notice that time courses of members of any particular cluster have similar patterns, as expected.

Before leaving our computational examples, we mention the open source software for bioinformatics available in BioConductor (<http://www.bioconductor.org>). In particular, this contains a number of powerful statistical tools for the analysis of microarray data.

11.5.2 Principal Components Analysis

In the example above, we were looking for shared patterns among *rows* of the gene expression matrix. As we indicated above, these shared patterns could be described by a correlation coefficient. It is often observed that there are correlations between *columns* of the gene expression matrix. In other words, there are correlations between the variables used to describe the gene expression data. For example, if x_i corresponds to expression levels or ratios for all genes at one point in the cell cycle and x_j corresponds to expression levels or ratios for these genes one generation time later, x_i may be correlated with x_j . When there are correlations between the experimental variables, it may be convenient to convert variables $\{x_i\}$ to a new set of variables $\{y_i\}$ that are uncorrelated. If the y_i are appropriately selected so that y_1 has the greatest variance and each successive y_i accounts for successively lesser amounts of the variance, it may be possible to represent the data with only the first d variables $y_1, y_2, \dots, y_d, d < m$, thus achieving data reduction. The gene expression data represented in terms of these new coordinates can then be clustered.

The procedure just described is known as **principal components analysis** (PCA). We do not discuss the mechanics of this process but will just give an idea of the approach, in effect restating more carefully the procedure described in the previous paragraph. For details, see Everitt and Dunn (2001). Principal components analysis defines m new variables or components $\{y_i\}$, with each new variable represented as a linear combination of the m original ones $\{x_i\}$:

$$y_i = \sum a_{ij}x_j.$$

This in effect *rotates the coordinate axes* so that the variation in the data occurs only with respect to these new axes, or components. As a consequence

of the transformation, the covariance matrix S (this time an $m \times m$ matrix whose elements are $\text{Cov}(x_i, x_j)$) is converted into a diagonal matrix A whose diagonal elements λ_{ii} are the variances corresponding to each component. The off-diagonal elements of A are all 0, indicating that with this transformation the newly defined component vectors are uncorrelated in the new coordinate frame of reference. A is constructed so that the diagonal elements become smaller as one goes from the first element at the top left to the last element at the bottom right. This means that λ_{11} represents the largest amount of variation in the data and λ_{mm} represents the least amount. The proportion of the total variation contributed by each component i is given by $\lambda_{ii} / \sum \lambda_{ii}$. Data reduction can be achieved by examining the diagonal elements and ignoring those components that contribute little to the total variation. For example, we might retain the set of d largest components that together represent 90% of the variation. Other components would be ignored because they do not contribute much to the variation among genes. An example of this approach to synchronized yeast cells has been presented by Alter et al. (2000). The final step is to perform clustering of the genes based upon similarity in their expression patterns described by the set of variables $y_i, i = 1, \dots, d$.

11.5.3 Confirmation of Results

Microarray experiments are often used to identify genes that should be examined further by extensive genetic, biochemical, or other “wet lab” approaches. Since these downstream experiments can involve many person-years of effort, it is essential to use independent methods to confirm differential expression of interesting genes identified by microarray experiments. This will help to eliminate the false positive identifications discussed in Section 11.4.2.

It is beyond the scope of this book to provide detail on this topic, but we briefly describe real-time (kinetic) PCR. For RNA samples, a reverse transcription (RT) step is employed, and in that case the method is called real-time RT-PCR. This method for quantifying DNA or RNA has a dynamic range extending over six orders of magnitude, and after PCR has begun it can be performed automatically without withdrawing samples from the reaction tube.

Recall that the amount of DNA produced by PCR after any number of cycles is proportional to the initial amount of the specific DNA or RNA present (Chapter 1.5.1). After the elongation step in each cycle, duplex DNA is formed, and this can be detected by enhanced fluorescence of dyes that insert between the stacked bases of the double helix (a process called intercalation). During the early stages of the PCR reaction, the amount of duplex doubles after every cycle, but for molecules at initially low concentrations, their fluorescence is insufficient to be detected above the background fluorescence in solution. Eventually, however, the concentration of the amplified molecules grows sufficiently for the fluorescence of the solution to begin increasing. The cycle number at which this occurs is called the *threshold cycle*.

DNA or RNA species present at initially low concentrations will require more cycles before they can be detected, and their threshold cycles will therefore be greater than is the case for molecules at high concentrations. Comparison of the threshold cycle number for gene X under treatment and control conditions with values for a standard curve will provide an independent and sensitive measure of the expression ratio. Commercial instruments are available for performing real-time PCR and automatically collecting kinetic data for 96 samples simultaneously. Methods for quantification other than the one described above are also available.

11.6 Examples of Experimental Applications

Before considering specific examples of applications, we should revisit the question of data structure. Recall that the data for experiments using cDNA or oligonucleotide microarrays are recorded in an expression matrix, often with rows corresponding to genes and columns corresponding to conditions. The ordering of the rows may be arbitrary, without any biological significance. We indicated above that hierarchical clustering can organize the rows by similarity of expression patterns (based upon distance measures in a high-dimensional vector space or on correlation coefficients). The columns (corresponding to different conditions) may or may not have a natural order or grouping. For example, if each column corresponds to a different specimen of a particular type of tumor, the order of columns may be arbitrary. Alternatively, if columns correspond to samples from a time course experiment, such as cells synchronized by the addition of a growth component previously withheld or embryos at different developmental stages, then interpretation of the results is facilitated if the conditions (columns) appear in temporal order. Sometimes data from several different time course experiments may be aggregated (e.g., expression patterns at different times after heat shock or the addition of a hormone), and the order of these groups of conditions may be arbitrary. It can be useful to cluster not only the genes but also the conditions (vector components). Clustering of vector *components* groups together conditions that have the same gene expression pattern. We illustrate these concepts with an artificial example before presenting examples of actual experimental results.

The goal is to present high-dimensional multivariate data in a manner that can easily be visualized. A conventional way of doing this is to present a color-coded gene expression matrix after clustering by rows or by rows and columns (Eisen et al., 1998). For example, degrees of red shading may indicate the extent of elevation in gene expression, degrees of green shading may represent the extent of reduction in gene expression, and black may signify no change relative to the reference state. This is illustrated by the simplified example in Fig. 11.9 (original data from Table 11.1). In this illustration, four different grey scale shadings are used to represent levels of gene expression for ten genes A, B, ..., J under ten conditions i, ii, \dots, x . In Fig. 11.9A, data have been

clustered by rows using the R applications `dist(,method="euclidean")` and `hclust(,method="average")` followed by `plclust()`. The pattern of shading allows us to recognize the underlying expression patterns immediately, in contrast with the effort that would be required for numerical comparisons of each component. We define three clusters of genes, each of whose members share similar expression patterns.

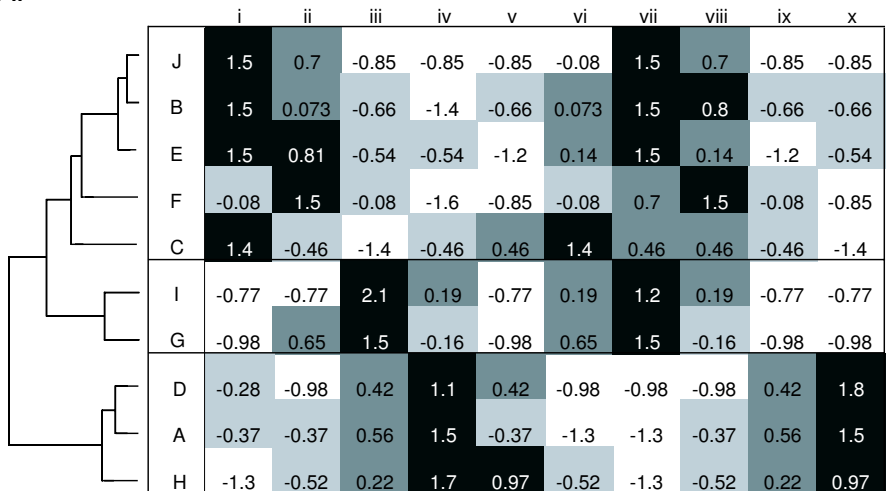
Table 11.1. Illustrative data for Fig. 11.9. Rows correspond to “genes” in alphabetical order, and columns correspond to conditions. Data have been scaled by rows using the means and standard deviations for each row.

	<i>i</i>	<i>ii</i>	<i>iii</i>	<i>iv</i>	<i>v</i>	<i>vi</i>	<i>vii</i>	<i>viii</i>	<i>ix</i>	<i>x</i>
A	-0.370	-0.370	0.560	1.50	-0.37	-1.300	-1.30	-0.37	0.560	1.50
B	1.500	0.073	-0.660	-1.40	-0.66	0.073	1.50	0.80	-0.660	-0.66
C	1.400	-0.460	-1.400	-0.46	0.46	1.400	0.46	0.46	-0.460	-1.40
D	-0.280	-0.980	0.420	1.10	0.42	-0.980	-0.98	-0.98	0.420	1.80
E	1.500	0.810	-0.540	-0.54	-1.20	0.140	1.50	0.14	-1.200	-0.54
F	-0.078	1.500	-0.078	-1.60	-0.85	-0.078	0.70	1.50	-0.078	-0.85
G	-0.980	0.650	1.500	-0.16	-0.98	0.650	1.50	-0.16	-0.980	-0.98
H	-1.300	-0.520	0.220	1.70	0.97	-0.520	-1.30	-0.52	0.220	0.97
I	-0.770	-0.770	2.100	0.19	-0.77	0.190	1.20	0.19	-0.770	-0.77
J	1.500	0.700	-0.850	-0.85	-0.85	-0.078	1.50	0.70	-0.850	-0.85

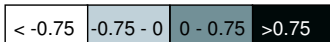
Figure 11.9B illustrates additional clustering by conditions. The goal here is to identify *groups of conditions* i, ii, \dots, x whose *genes* show similar expression patterns. We achieve this clustering by transposing the expression matrix (after rows were reordered) and applying the same R functions as we used to produce Fig. 11.9A. If the conditions were time points during the cell cycle, then we would expect that time points for successive iterations of each stage of the cell cycle would cluster together. If the conditions corresponded to samples from tumors, we would expect like tumor types to be grouped together. If

Fig. 11.9 (opposite page). Graphical representation and clustering of gene expression data. Panel A: Expression data for genes A, . . . , J in Table 11.1 were subjected to hierarchical clustering as described in the text, and results are displayed as described by Iyer et al. (1999), except for the use of a grey scale instead of color. Expression levels indicated by each shade are indicated by the strip between panels A and B. Horizontal lines separate three clusters based upon the dendrogram shown at the left. Panel B: Same data as in panel A, except that the conditions i, ii, \dots, x have now also been clustered based upon shared patterns of gene expression. Two major clusters are seen (dendrogram at top and graphic), each with a characteristic gene expression pattern (*iii* appears as a singleton in this example).

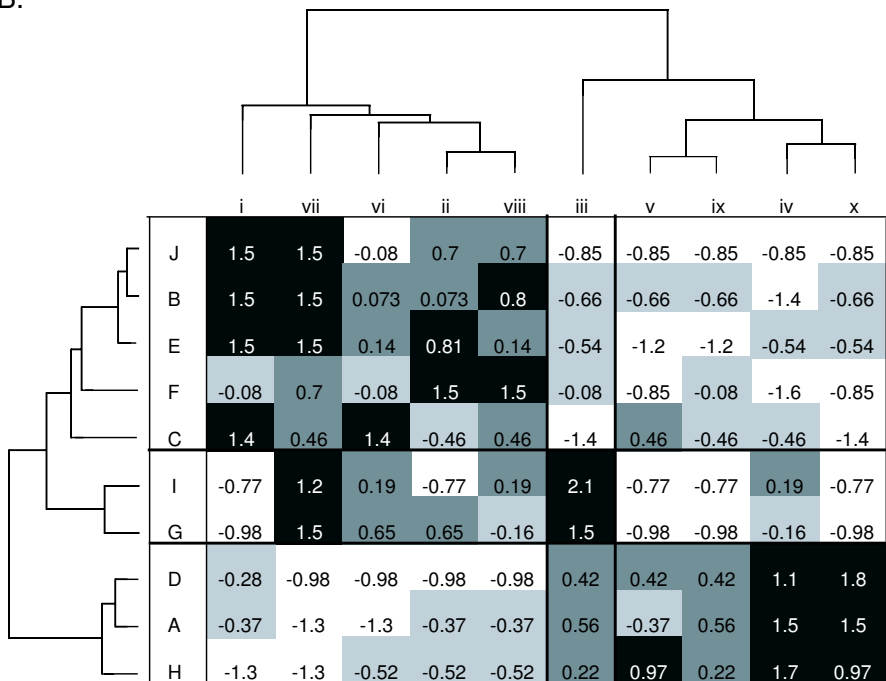
A.



Scaled Expression



B.



the conditions were samples from different species of animals, we might expect that conditions would form clusters that reflect the phylogenetic relationships of those animals. In this example, we see that there are two major groups of conditions and a singleton.

This simplified introduction sets the stage for examples of gene expression results. There are thousands of studies reported or in progress, and we discuss only four to illustrate the type and utility of such experiments.

11.6.1 Gene Expression in Human Fibroblasts

Animal cells growing in tissue culture ordinarily require growth factors. These are often supplied by adding serum to the growth medium. Human fibroblast cells isolated from foreskins were cultured in a medium lacking serum, and after 48 hours serum was restored (Iyer et al., 1999). The cells thus synchronized with respect to phase in the cell cycle resumed growth, and the expression of 8600 human genes was assessed as a function of time over a 24 hour period. Genes having like expression patterns were clustered to identify the stages of the cell cycle during which their expression was elevated or reduced.

Figure 11.10 shows a portion of the clustered data (Eisen et al., 1998). Gene annotations identified the functions of the genes in the clusters illustrated. Cluster B (16 genes) represented genes involved in the cell cycle, and Cluster C (9 genes) consisted of genes involved in the immediate-early response (e.g., transcription factors). The commonalities in expression patterns are readily understood from this type of graphical presentation. In this case, no clustering of conditions was required because the sampling times correspond to successive stages in the cell cycle, and only one cell cycle was monitored.

11.6.2 Gene Expression During *Drosophila* Development

The developmental program of *Drosophila* follows a typical insect progression from embryo (E) through larval (L), pupal (P), and adult (A) stages. Understanding the suites of genes that are coordinately expressed is crucial to understanding the genetic networks controlling development in insects and other metazoan animals. Accordingly, expression patterns of 4028 genes were measured over the entire *Drosophila* life cycle (Arbeitman et al., 2002).

Clustering of patterns for all 4028 genes was performed based upon their individual expression patterns at successive developmental time points (clustering by rows; Fig. 11.9A). Members of the resulting clusters of genes were often functionally related. For example, one cluster was enriched for genes active in terminally differentiated muscle (Fig. 11.11). DNA sequences of these coexpressed genes were analyzed for the presence of binding sites for transcription factors known to be involved in muscle differentiation. The combined data revealed within this cluster genes whose involvement in muscle terminal differentiation had not previously been recognized. Expression of terminally differentiated muscle tissue genes is particularly high in the late embryo/larval

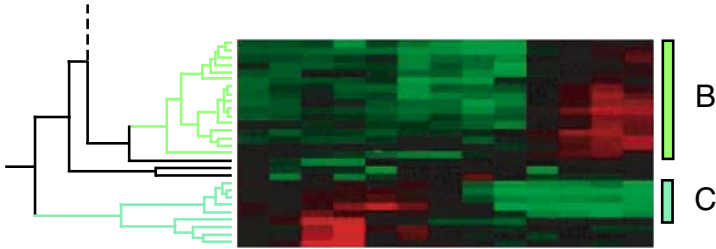


Fig. 11.10. [This figure also appears in the color insert.] Expression of clusters of genes from serum-deprived fibroblasts for 12 time points after serum is restored. Each horizontal strip corresponds to the expression profile of one gene. Red or green, respectively, indicate elevation or depression of gene activity compared with levels in serum-deprived cells. Portions of the dendrogram resulting from clustering of the entire set of 8600 genes are indicated at the left. Cluster B includes genes involved in the cell cycle, and cluster C corresponds to genes involved in the immediate-early response. Excerpted, with permission, from Eisen MB et al. (1998) *Proceedings of the National Academy of Sciences USA* 95:14863–14868. Copyright 1998 National Academy of Sciences USA.

stage and at the end of the pupal stage. Note the similarity in patterns reading down the last few columns in stages E and P in Fig. 11.11.

The two “waves” of elevated gene expression evident at the end of the embryonic and pupal stages illustrate for this particular gene cluster the general result obtained from *all* 4028 genes clustered by columns (Fig. 11.9B). For such an analysis, vectors for each time point (containing expression levels at that time point for all *genes*) were clustered based on similarity of expression patterns (illustration not shown). Two large clusters were observed: time points from the larval stage clustered with time points associated with adults, and time points associated with the embryo clustered with time points from the pupal stage. This indicates that similar developmental circuits employing the same genes may be involved at different stages of the developmental process.

11.6.3 Gene Expression in Diffuse Large B-cell Lymphomas

Diffuse large B-cell lymphomas are malignancies associated with B-lymphocytes (antibody-producing cells of the immune system). Expression profiles for genes corresponding to more than 20,000 cDNA clones were obtained for tissue samples taken from 44 human biopsies and were compared with a variety of normal and malignant control samples (Alizadeh et al., 2000). The B-cell lymphoma samples showed distinctive expression patterns compared with control samples. Clustering of genes based on their expression patterns was performed, and then samples were clustered using a subset of the genes tested. Gene expression patterns for the biopsy samples fell into two distinct

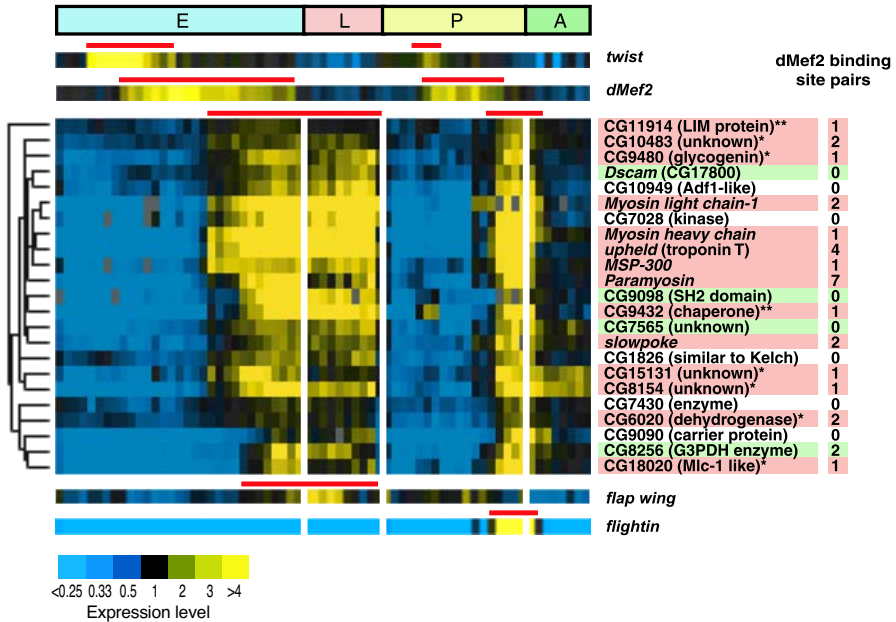


Fig. 11.11. [This figure also appears in the color insert.] Expression of terminally differentiated muscle genes during *Drosophila* development. Graphical conventions are similar to Fig. 11.10, except that yellow indicates high levels of expression and blue indicates low levels of expression. The multicolored ribbon at the top spans the times corresponding to embryonic (E), larval (L), pupal (P), or adult (A) stages of development. Expression patterns for transcription factors *twist* and *dMef2*, known regulators of muscle development, are indicated immediately below the developmental stage indicator. Their expression prior to the onset of expression of muscle genes was anticipated. Reprinted, with permission, from Arbeitman MN et al. (2002) *Science* 297:2270-2275. Copyright 2002 American Association for the Advancement of Science.

classes: those with characteristic patterns of B-cell germinal centers or those similar to activated B-cells.

The clinical outcomes for patients treated by multiagent chemotherapy were monitored and compared with the gene expression patterns of the corresponding biopsy samples. Those patients having lymphomas with expression patterns like germinal center B-cells had substantially higher 5 year survival rates (76%) than did patients having lymphomas with the activated B-cell patterns (16%). This expression study identified subclasses of B-cell lymphomas that had not previously been recognized, and the expression patterns of these subclasses were correlated with distinct clinical outcomes.

11.6.4 Analysis of the Yeast Transcriptome Using SAGE

SAGE was used to analyze the *S. cerevisiae* transcriptome at three different stages of growth (Velculescu et al., 1997). The 60,000 sequenced SAGE tags provided estimates of the fractional amounts of each transcript per cell for transcripts present at levels higher than 0.3 transcripts/cell. These fractions, together with prior estimates that yeast contains 15,000 mRNA molecules, indicated that the majority of genes were represented in the transcriptome at levels of about 1 to 2 mRNA molecules per cell. Indeed, 55% of the total number of transcripts came from just 160 of the 4665 genes whose transcripts were detected. To detect transcripts present at very low levels, much higher levels of “coverage” (number of tags sequenced/mRNA molecules present) are needed. The advantages of an open architecture (variables not predetermined) were realized since transcripts corresponding to 160 ORFs that had not been annotated in the genome were detected. Comparison of expression patterns at the different growth states indicated that less than 1% of the genes were differentially expressed. (Using time course data and oligonucleotide arrays, Cho et al. (1998) found that 6.7% of the yeast genes were differentially regulated over the course of the cell cycle.) These results confirm the assertion that usually most genes are not differentially expressed (see Section 11.4.1).

11.7 Protein Expression

Because transcript levels may not accurately represent the concentrations of their cognate proteins (Section 11.1), direct measurement of protein expression is desirable. Much current proteome research is directed toward technology development, so computational and statistical issues have been emphasized to a lesser extent than has been the case for oligonucleotide- or cDNA-microarray (collectively, DNA microarray) experiments. For this reason, we provide only a brief summary of some of the experimental approaches for measuring protein expression levels.

Studies of the protein complement in cells may be directed toward several different goals:

- Discovering which known proteins are coordinately regulated;
- Characterizing the abundances of different proteins in particular cells or cell compartments;
- Comparing protein levels in cells from diseased individuals or in cells treated with a pharmaceutical agent with protein levels in normal (control) cells;
- Determining what ligands are bound by different proteins;
- Identifying partners in protein-protein interactions.

Notice that the first three items in this list are similar to the goals for studies with DNA microarrays. The last two items are new and are two of the goals of *functional proteomics*.

Detecting protein species over the wide range of concentrations at which they may appear in the cell is a significant technical problem. With simple organisms such as yeast, in-frame fusions of DNA encoding an affinity tag can be placed at the end of each ORF in the genome by recombinant DNA techniques (Ghaemmaghami et al., 2003). This allows a single convenient purification protocol to be used for all proteins in the genome and also facilitates their detection by highly sensitive antibody techniques. Such methods revealed about 4250 protein species in yeast grown in log-phase (about 75% of the estimated total number of genes). In contrast, only 500–1500 yeast protein species were detected using liquid chromatography/mass-spectrometry (LC/MS) methods (see below). 2D gel electrophoresis and LC/MS methods also fail to detect a large proportion of low-abundance proteins. For example, only 8% of yeast proteins appearing at 5000 copies per cell or less were detected by MS methods (Ghaemmaghami et al., 2003). The less sensitive methods for protein detection are still employed because in-frame fusion of affinity tags is technically more difficult for complex eukaryotic genomes.

11.7.1 2DE/MALDI-MS

The older, well-established two-dimensional electrophoresis (**2DE**) method has been joined to matrix-assisted laser-induced desorption/ionization (**MALDI**) mass spectrometry (**MS**). There are two steps to this composite procedure: resolution of polypeptides on two-dimensional polyacrylamide gels (Section 1.5) and identification of each spot by mass spectrometry. Each well-resolved spot on the 2D gel corresponds (ideally) to a single polypeptide characterized by a molecular mass and a pI. These parameters alone are not sufficient to *identify* the actual polypeptide chains. Therefore, spots are excised from the gel, digested with a sequence-specific protease such as trypsin, and then subjected to MALDI-MS.

Because this is a highly technical subject beyond the scope of this book, we only provide a brief outline of the principles of MALDI-MS. There are three principal instrumental components: an ionization source, a mass analyzer, and a detector. With MALDI, the digested polypeptide sample is embedded in an organic matrix, that is then dried and placed in a vacuum. Pulses from a UV laser volatilize the sample. The molecules may become charged by acquiring one or more H^+ ions in the process. These charged polypeptides (now in the gas phase) are accelerated toward the mass analyzer by application of a high-voltage electric field (e.g., 25 kV). The mass analyzer employs a magnetic field to resolve the individual polypeptides. Moving charges in a magnetic field experience a force proportional to the charge and perpendicular to both the velocity vector *and* to the magnetic field vector. Thus the ion trajectories in the mass analyzer become broad arcs. A single peptide type may produce one or more species having different charges, and each species follows a different trajectory. Different peptide fragments having the same charge differ in mass, and they also follow different trajectories because they were accelerated to

different velocities. In general, peptide fragments having different mass (m) to charge (z) ratios, m/z , can be resolved by the analyzer. The detector measures the ion current at different values of m/z . The overall precision in resolving species of different m/z is 10 ppm (!). Each particular m/z value usually corresponds to a single polypeptide amino acid sequence, which identifies the polypeptide.

Greater accuracy in polypeptide identification can be achieved by tandem mass spectrometry. In this case, two components are added between the mass analyzer and the detector: a collision cell and a second mass analyzer. Individual polypeptides from the first mass analyzer are directed into the collision cell, which contains a low concentration of the inert gas argon (Ar). High-speed collision of the polypeptide with the argon atoms causes collision-induced dissociation (CID). The resulting polypeptide *fragments* are then resolved by the second mass analyzer. The m/z values of the constituent fragments provide a fingerprint of the polypeptide.

This whole operation is obviously *not* a high-throughput, parallel analytical procedure. Efforts to improve this general approach substitute tandem microcapillary chromatography for the 2DE (e.g., strong cation-exchange chromatography followed by reverse-phase chromatography). The effluent from the second column is then analyzed by MS (see Ferguson and Smith, 2003, for a review). Obviously, it would be desirable to have analytical methods that offer some of the same convenience as DNA microarrays.

11.7.2 Protein Microarrays

Protein microarrays employ probes that specifically recognize different target proteins in a complex mixture of target species. One of the goals might be simply quantification of the *amounts* of the different protein species (analogous to DNA microarrays). Other goals might be detection of protein-protein interactions, protein-small molecule interactions, or enzyme-substrate interactions. There are two general types of protein microarrays: (1) arrays of the *proteins* of interest and (2) arrays of *antibodies or antibody-like molecules* directed against the proteins of interest. If the proteins themselves are arrayed as probes on solid substrates, they may lose biochemical activity. Also, different arrayed proteins may have different pH or ionic strength optima. Any one condition is unlikely to be optimal for all proteins on the array. These issues are less important if the probes are antibodies directed against specific target molecules. An advantage of arrayed antibodies is that these proteins are more robust and comparably active under a particular assay condition. However, even for antibodies, instability on surfaces or during long-term storage may be problematic.

We limit our discussion of arrayed antibody probes (see MacBeath, 2002, for a review). Suppose that we had a collection of antibody samples Ab_a , Ab_b , \dots , Ab_z , each capable of recognizing antigens a , b , \dots , z . We could imagine using these antibodies to create an **antibody microarray** for detecting

and quantifying antigens a, b, \dots, z that might be present in a protein sample of interest. Given the necessary antibody probes, the array technology is relatively straightforward and analogous to DNA microarray technology. The probes (sometimes called *capture antibodies*) are robotically arrayed and mapped onto a suitably prepared glass slide and exposed to a target solution containing the proteins (antigens) to be analyzed. The antigens in the sample specifically bind to the probes or capture antibodies, and features on the array are then scanned to detect bound antigens.

One method to detect antigen binding is indirect labeling by detection antibodies (Fig. 11.12A). The detection antibodies recognize the same antigens as the set of probes or capture antibodies, but they recognize a different epitope. When a cocktail of detection antibodies is placed on an antibody array that has been incubated with the target sample, an antibody-antigen-antibody “sandwich” is formed at those features that contain bound antigen. The detection antibodies either contain fluorescent labels or chemical groups that allow them to act as reporter molecules. This method obviously requires two antibody preparations for every antigen to be assayed: one to be arrayed as the probe and one for detection.

A second method (and the one employed with commercial arrays having large numbers of features) is the *antigen capture* method, which is analogous to the method used with DNA spotted microarrays (Fig. 11.12B). With this method, there are two target samples: treatment and control. Each target sample is labeled with a different fluorescent dye (Cy3 or Cy5, for example), the samples are mixed, and then they are incubated with the probes on the antibody microarray. The relative amounts of each antigen in the two samples are determined from the ratios of the fluorescence intensities for the two dyes. Commercially available arrays (BD Biosciences, Palo Alto, CA) may contain 500 different monoclonal antibodies directed against a variety of human proteins (transcription factors, receptors, cell cycle proteins, cytoskeletal proteins, membrane proteins, and several other classes of proteins). The arrays and reagent kits are designed for comparing two protein samples labeled with either Cy3 or Cy5. The array format is compatible with the same instrumentation used for DNA spotted microarrays. Two different slides are supplied for use with a dye-swap protocol. If T is the treatment sample and C is the control sample, four different labeled samples are produced: T-Cy3, T-Cy5, C-Cy3, and C-Cy5. One slide is incubated with a mixture of T-Cy5+C-Cy3, and the other is incubated with a mixture of T-Cy3+C-Cy5. Each feature on each slide appears in duplicate. To study other organisms, or to include antibodies directed against any of the other 25,000–75,000 human proteins and their in vivo modified variants, the particular antibody probes must be produced by the investigator or by a commercial vendor. Alternatives to antibody probes can be produced by phage display technologies or by protein engineering, but these will not be discussed here.

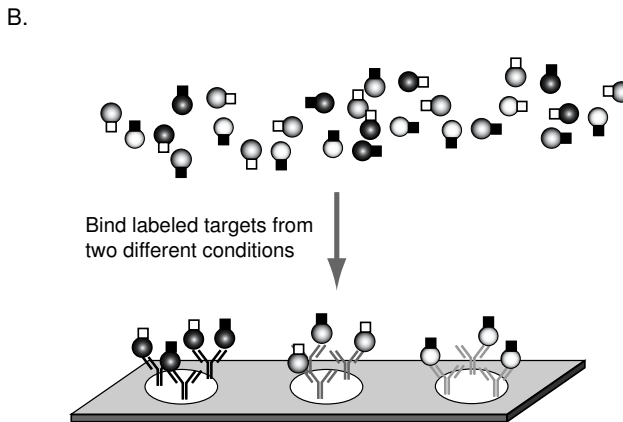
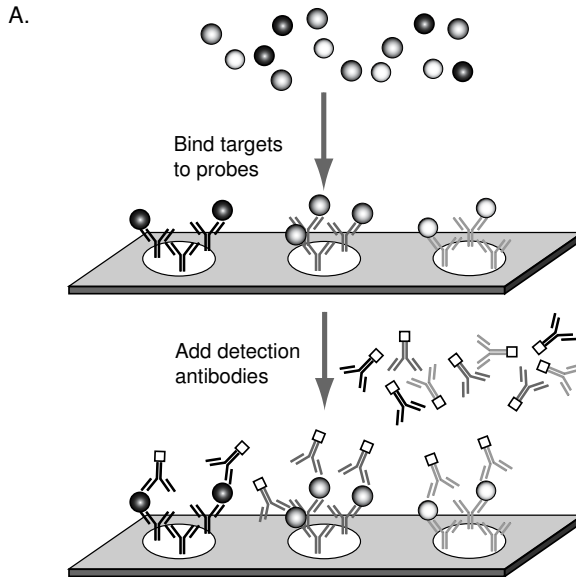


Fig. 11.12. Schematic illustration of antibody microarrays. Each feature on the slide corresponds to an antibody (Y-shaped patterns) directed to a different protein antigen (shaded spheres). Antigens and cognate antibodies have similar shadings. Panel A: Bound antigens are quantified by indirect labeling using a mixture of detection antibodies, all labeled with the same dye (open squares). Antigen becomes sandwiched between the capture antibodies constituting each feature and the detection antibodies. Panel B: Antigen capture format. Target proteins are isolated from two conditions and differentially labeled with two different fluorescent dyes (open and filled squares). After binding to the antibody features, amounts of antigen corresponding to each condition can be quantified by measuring fluorescence intensity in two wavelength channels.

11.8 The End of the Beginning

We have presented in this chapter the basics of measuring the levels of mRNA and proteins in cells. The methods that we have discussed in-depth are multiplex approaches that measure responses of a large number of genes or gene products to a variety of experimental conditions. The data are intrinsically multivariate, with individual vectors (expression patterns of one gene) potentially having 10 to 100 components or more. We indicated how clustering methods could be employed to produce a cluster of genes that display similar patterns of expression under different experimental conditions or times in the cell cycle. We close this chapter by discussing the uses of such data.

Cells and organisms are structurally complex entities that exhibit a large repertoire of responses to changes in their environment. We would not claim complete understanding of a biological system unless it were possible to trace all changes in protein and substrate concentrations and all structural changes of a cell or organism in response to any outside perturbation (e.g., temperature shift, change in hormone concentration, amino acid starvation). There are about 10^4 to 10^5 genes in genomes of multicellular eukaryotes, and for those gene products that are enzymes, there are collectively thousands of products and substrates. The functions of proteins in regulatory and metabolic pathways are highly coordinated, forming **networks** of functionally related proteins. The biochemical system is not at equilibrium. Rather, there are directional flows of energy and reaction products through this network of related processes. How can we describe such a complex system?

At a minimum, the description of a living cell requires a parts inventory, a description of the concentrations of all molecules and substrates as a function of times and conditions, and a specification of how the functions of the individual components (molecules) are coordinated. Since living cells are dynamic, nonequilibrium systems, we could include rate constants for all biochemical reactions. This type of integrative approach falls within the purview of **systems biology**. It contrasts with the typical **reductionist approaches** (studies of single molecules, operons, or pathways) that have been so successful in molecular biology over the last half century.

Gene expression analysis (measurement of mRNA or protein concentrations) provides basic initial information for an eventual integrated view of cell function and behavior. These measurements indicate which genes are expressed under each experimental condition, and clustering methods suggest which sets of genes may be coordinately regulated. For coordinately-expressed genes, we can compare sequences of promoter regions in an attempt to identify transcription factor binding sites that might be fundamental to this process. Alternatively, protein products of coordinately expressed genes might be tested directly for protein-protein interactions. As we already indicated, expression analysis can be used to suggest functions for genes not yet annotated, and functional annotation is an essential part of placing each gene product into the appropriate mechanistic context.

The solution of the problems of systems biology will not come from bioinformatics alone, or even from current expression array data in conjunction with bioinformatics. While these initial steps are important, systems biology will require substantial new biotechnology methods and new types of informatics.

References

- Alizadeh AA, Eisen MB, Davis RE, Ma C, Lossos IS, Rosenwald A, Boldrick JC, Sabet H, Tran T, Yu X, Powell JI, Yang L, Marti GE, Moore T, Hudson J Jr, Lu L, Lewis DB, Tibshirani R, Sherlock G, Chan WC, Greiner TC, Weisenburger DD, Armitage JO, Warnke R, Levy R, Wilson W, Grever MR, Byrd JC, Botstein D, Brown PO, Staudt LM (2000) Distinct types of diffuse large B-cell lymphoma identified by gene expression profiling. *Nature* 403:503–511.
- Alter O, Brown PO, Botstein D (2000) Singular value decomposition for genome-wide expression data processing and modeling. *Proceedings of the National Academy of Sciences USA* 97:10101–10106.
- Arbeitman MN, Fleming AA, Siegal ML, Null BH, Baker BS (2004) A genomic analysis of *Drosophila* somatic sexual differentiation and its regulation. *Development*, 131:2007–2021.
- Arbeitman MN, Furlong EEM, Imam F, Johnson E, Null BH, Baker BS, Krasnow MA, Scott MP, Davis RW, White KP (2002) Gene expression during the life cycle of *Drosophila melanogaster*. *Science* 297:2270–2275.
- Arfin SM, Long AD, Ito ET, Tollerli L, Riehle MM, Paegle ES, Hatfield GW (2000) Global gene expression profiling in *Escherichia coli* K12. *Journal of Biological Chemistry* 275:29672–29684.
- Cho RJ, Campbell MJ, Winzeler EA, Steinmetz L, Conway A, Wodicka L, Wolfsberg TG, Gabrielian AE, Landsman D, Lockhart DJ, Davis RW (1998) A genome-wide transcriptional analysis of the mitotic cell cycle. *Molecular Cell* 2:65–73.
- Churchill GA (2002) Fundamentals of experimental design for cDNA microarrays. *Nature Genetics Supplement* 32:490–495.
- Eisen MB, Spellman PT, Brown PO, Botstein D (1998) Cluster analysis and display of genome-wide expression patterns. *Proceedings of the National Academy of Sciences USA* 95:14863–14868.
- Everitt BS, Dunn G (2001) *Applied Multivariate Data Analysis*. 2nd Ed. Oxford: Oxford University Press.
- Ferguson PL, Smith RD (2003) Proteome analysis by mass spectrometry. *Annual Review of Biophysics and Biomolecular Structure* 32:399–424.
- Futcher B, Latter GI, Monardo P, McLaughlin CS, Garrels JI (1999) A sampling of the yeast proteome. *Molecular and Cell Biology* 19:7357–7368.

- Ghaemmaghami S, Huh W-K, Bower K, Howson RW, Belle A, Dephoure N, O'Shea EK, Weissman JS (2003) Global analysis of protein expression in yeast. *Nature* 425:737–741.
- Gygi SP, Rochon Y, Franza BR, Aebersold R (1999) Correlation between protein and mRNA abundance in yeast. *Molecular and Cell Biology* 19:1720–1730.
- Iyer VR, Eisen MB, Ross DT, Schuler G, Moore T, Lee JCF, Trent JM, Staudt LM, Hudson J Jr, Boguski MS, Lashkari D, Shalon D, Botstein D, Brown PO (1999) The transcriptional program in the response of human fibroblasts to serum. *Science* 283:83–87.
- Long AD, Mangalam HJ, Chan BYP, Tollerli L, Hatfield GW, Baldi P (2001) Improved statistical inference from DNA microarray data using analysis of variance and a Bayesian statistical framework. *Journal of Biological Chemistry* 276:19937–19944.
- MacBeath G (2002) Protein microarrays and proteomics. *Nature Genetics Supplement* 32:526–532.
- Sutcliffe JG, Foye PE, Erlander MG, Hilbush BS, Bodzin LJ, Durham JT, Hasel KW (2000) TOGA: An automated parsing technology for analyzing expression of nearly all genes. *Proceedings of the National Academy of Sciences USA* 97:1976–1981.
- Tan PK, Downey TJ, Spitznagel EL Jr, Xu P, Fu D, Dimitrov DS, Lempicki RA, Raaka BM, Cam MC (2003) Evaluation of gene expression measurements from commercial microarray platforms. *Nucleic Acids Research* 31:5676–84.
- Velculescu VE, Zhang L, Vogelstein B, Kinzler KW (1995) Serial analysis of gene expression. *Science* 270:484–487.
- Velculescu VE, Zhang L, Zhou W, Vogelstein J, Basrai MA, Bassett DE Jr, Hieter P, Vogelstein B, Kinzler KW (1997) Characterization of the yeast transcriptome. *Cell* 88:243–251.
- Yang YH, Dudoit S, Luu P, Lin DM, Peng V, Ngai J, Speed TP (2002) Normalization for cDNA microarray data: A robust composite method addressing single and multiple slide systematic variation. *Nucleic Acids Research* 30: e15.
- Yang YH, Speed TP (2002) Design issues for cDNA microarray experiments. *Nature Reviews Genetics* 3:579–588.

Exercises

Exercise 1. Spotted microarrays allow measurement of fluorescence intensities corresponding to mRNA levels for each gene or feature.

- a. Are intensity ratio measurements for low-abundance transcripts independent of measurements for high-abundance transcripts when reverse-transcribed DNA from the same mRNA sample is used for the hybridization?

- b. Is there a similar dependence or independence for transcripts whose abundance is measured by SAGE?

Exercise 2. Suppose that SAGE is used to analyze gene expression from a eukaryote having 30,000 genes each expressed at a level of two transcripts per cell on average. If the average number of ditags per cloned insert is 20, what is the least number of clones that must be sequenced to detect, with probability 0.95, gene expression at a level of 0.5 transcripts/cell? [Hint: This is a coverage problem. See Section 4.5.]

Exercise 3. Perform hierarchical clustering of the yeast data in Section C.4. Use the Euclidean distance metric with standardized data. How does cluster membership obtained with hierarchical clustering compare with the result from K -means (Computational Example 11.2)?

Exercise 4. Repeat the calculation in Exercise 3, except use the correlation coefficients to calculate distances. [Hint: Use the R function `as.dist()`.] In previous calculations using these data, we standardized expression levels for each *gene* (row) prior to clustering. Is this standardization needed for hierarchical clustering when correlation coefficients are used for distances? Why or why not?

Exercise 5. Use the results from the example presented in Computational Example 11.2 and similar results obtained by you for $k = 5$ and $k = 6$ to produce a plot like Fig. 10.8.

Exercise 6. Examine the output for K -means, $k = 3$ in Computational Example 11.2. Suppose that you wanted to perform a *classification* of a set of unknown vectors based upon their similarity to the average pattern typical of Cluster 2.

- What *pattern vector* could you use to represent Cluster 2?
- For classification, you would need to produce a *score* that reports how close any candidate vector is to the pattern vector. What function discussed in Section 11.5 could be used to produce this score?
- Compute scores for previously defined members of Cluster 2 relative to the pattern vector representing Cluster 2 as a whole.
- Score previously defined members of Cluster 1 to test how well they conform to the pattern of Cluster 2.

Exercise 7. Intensity measurements from a single slide (Fig. 11.5) for four replicated features corresponding with the *twist* gene are presented below. Nucleic acid corresponding with *dsd*^D flies was labeled with Cy5 (R), and nucleic acid corresponding with wild-type flies was labeled with Cy3 (G).

Feature number	Intensity ^a at 635 nm	Intensity ^a at 532 nm
1175	1125	1683
2329	819	1621
3407	273	532
5717	1420	1888

^a Intensity values after subtraction of background intensity

- Correct the data using the global normalization factor obtained in Computational Example 11.1.
- What is the probability that the expression levels of *twist* in *dsd^D* flies differs from expression levels of *twist* in normal flies?
- What is the probability that $R/G > 2$?

Exercise 8. Use the data corresponding with Fig. 11.6 to estimate the fraction of features for which $|\log_2(R/G)| > 1$. If the standard errors of the intensities for *twist* (Exercise 7) are typical of the average feature, is the estimated fraction a good indicator of the number of genes whose expression differs between mutant and wild-type flies?

Exercise 9. Yeast cells contain approximately 15,000 transcript molecules per cell. SAGE analysis indicated that there are roughly three broad abundance classes of transcripts, as indicated below (Velculescu et al., 1997):

Component	Fraction of total RNA	Copies/Cell
1	0.17	180
2	0.35	40
3	0.45	2.5

Approximately how many genes are responsible for Component 1? For Component 1 + Component 2?

Exercise 10. For the *Drosophila* Cy3 data at <http://www.cmb.usc.edu>, compute the fraction of the total signal represented by each gene (after removing controls, blanks, and flagged data). Sort the fractions in order of increasing fraction, and then plot the cumulative fraction of RNA as a function of the fraction of genes examined. What fraction of these genes accounts for the most abundant 20% of the transcripts? Is there evidence for classes of genes distinguished by their expression levels?

Inferring the Past: Phylogenetic Trees

12.1 The Biological Problem

This chapter discusses methods for analyzing and describing ancestor-descendant relationships among groups of organisms. These groups may be formalized groupings recognized by biological classification systems (species, genus, subfamily, etc.) or they may be different populations within a species. Recognized groups are called **taxa** (singular: taxon). The actual ancestor-descendant relationships for a particular set of taxa are called a **phylogeny**. An example was shown in Fig. 1.1 in Chapter 1.

The actual phylogeny ordinarily is not known because ancestral organisms may have become extinct long before modern humans evolved to observe them. Therefore, phylogeny is inferred from data derived from organisms alive today or represented in the fossil record. In this chapter, we assume we are using molecular data. Phylogenetic relationships are of considerable historical and practical interest, and we first offer two specific examples that illustrate what phylogenies are and why they are important. Then we present an informal discussion of how to “read” trees before proceeding to quantitative methods for inferring them.

There is some overlap with our discussion of clustering in Chapter 10. In that chapter, we employed distances to create hierarchical clusters using either morphological or sequence data. In the example of clustering of primates based upon sequence data, we employed the number of sites at which two DNA sequences differ (i.e., the **Hamming distance**) to create a hierarchical cluster linking human, chimpanzee, gorilla, and gibbon. How does the construction of phylogenetic trees differ from clustering? First, ancestor-descendant relationships are explicitly invoked in tree-building. In contrast, points where OTUs join in hierarchical clustering need not correspond to hypothetical ancestral OTUs. Second, evolutionary mechanisms suggest that the distances between two OTUs in a tree should be divided between the two branches leading back to their common ancestor. With hierarchical clustering, the junction point was mapped to the corresponding distance, and the distance

was not divided between the two branches. Finally, with tree construction we recognize that evolutionary processes may not be adequately reflected in the distances that are directly measured, and accordingly appropriate corrections are applied. Hierarchical clustering employs the distances calculated directly from the measured characters. Despite these differences, there are obvious operational and conceptual connections between hierarchical clustering and the building of trees.

12.1.1 Example: Relationships Among HIV Strains

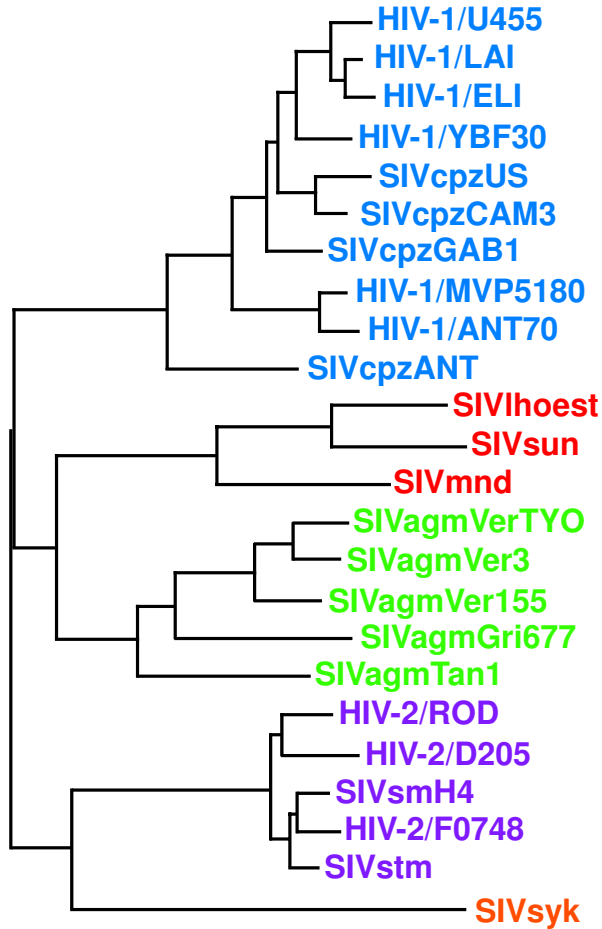
Human immunodeficiency viruses HIV-1 and HIV-2 are the causative agents for the AIDS pandemic (HIV-1 being more prevalent). Because HIV-1 is a *retrovirus* (replicating by reverse transcription of an RNA molecule), it accumulates mutations more rapidly than does human genomic DNA. The sequence variation that results can be revealed by sequencing DNA copies of the retroviral RNA molecules.

Populations of the variant HIV-1 strains constitute the groups and subtypes of the virus found among human populations today. For example, group M, subtype B is common among infected persons in the United States, while group M, subtype C is more common in Southeast Asia. By examining the sequences of HIV subtypes, it is possible to track the pattern of transmission of the viruses around the world. Figure 12.1 shows the relationship between HIV strains and various simian immunodeficiency viruses (SIV). It is seen that HIV-1 types are more closely related to SIVcpz than they are to HIV-2, which itself is more closely related to SIVs isolated from the sooty mangabey and the stump-tailed macaque. From these and similar data we can make the reasonable inference that the ancestors of the HIVs present today entered the human population by transmission of simian immunodeficiency viruses found in populations of chimpanzees and other primates. It is thus possible to infer the origins and date of inception of the pandemic (Hahn, 2000).

12.1.2 Example: Relationships Among Human Populations

We humans are keenly interested in our own origins. Although people from different regions of the world may differ substantially in appearance (e.g., a Masai from Africa compared with a Dane from Northern Europe), we all belong to the same species. How and where did the various human populations originate? This question can be approached by sequencing DNA from individuals belonging to many human populations from all over the globe. Sometimes chromosomal DNA (particularly X or Y chromosomal DNA) is employed, and sometimes mitochondrial DNA (mtDNA) is used.

Mitochondrial DNA of humans is a circular molecule containing some 16,500 bp. Over time, this DNA accumulates mutations (base changes) within its genome, and these mutations are passed on to descendants by the mother. (Mitochondrial DNA is maternally inherited.) Populations of humans that



0.10

Fig. 12.1. [This figure also appears in the color insert.] Examples of phylogenetic trees. Inferred relationships between human immunodeficiency virus (HIV) and simian immunodeficiency virus (SIV) strains isolated from humans and other primates: agm, African green monkey; cpz, chimpanzee; lhoest, L’Hoest monkey; mnd, mandrill; sm, sooty mangabey; stm, stump-tailed macaque; sun, sun monkey; syk, Sykes monkey. The tree was constructed from viral sequence data but may be confounded by recombination between different virus strains. Reprinted, with permission, from Hahn BH et al. (2000) *Science* 287:607–614. Copyright 2000 American Association for the Advancement of Science.

share particular sets of mutations are understood to be more closely related to each other than they are to populations lacking these mutations.

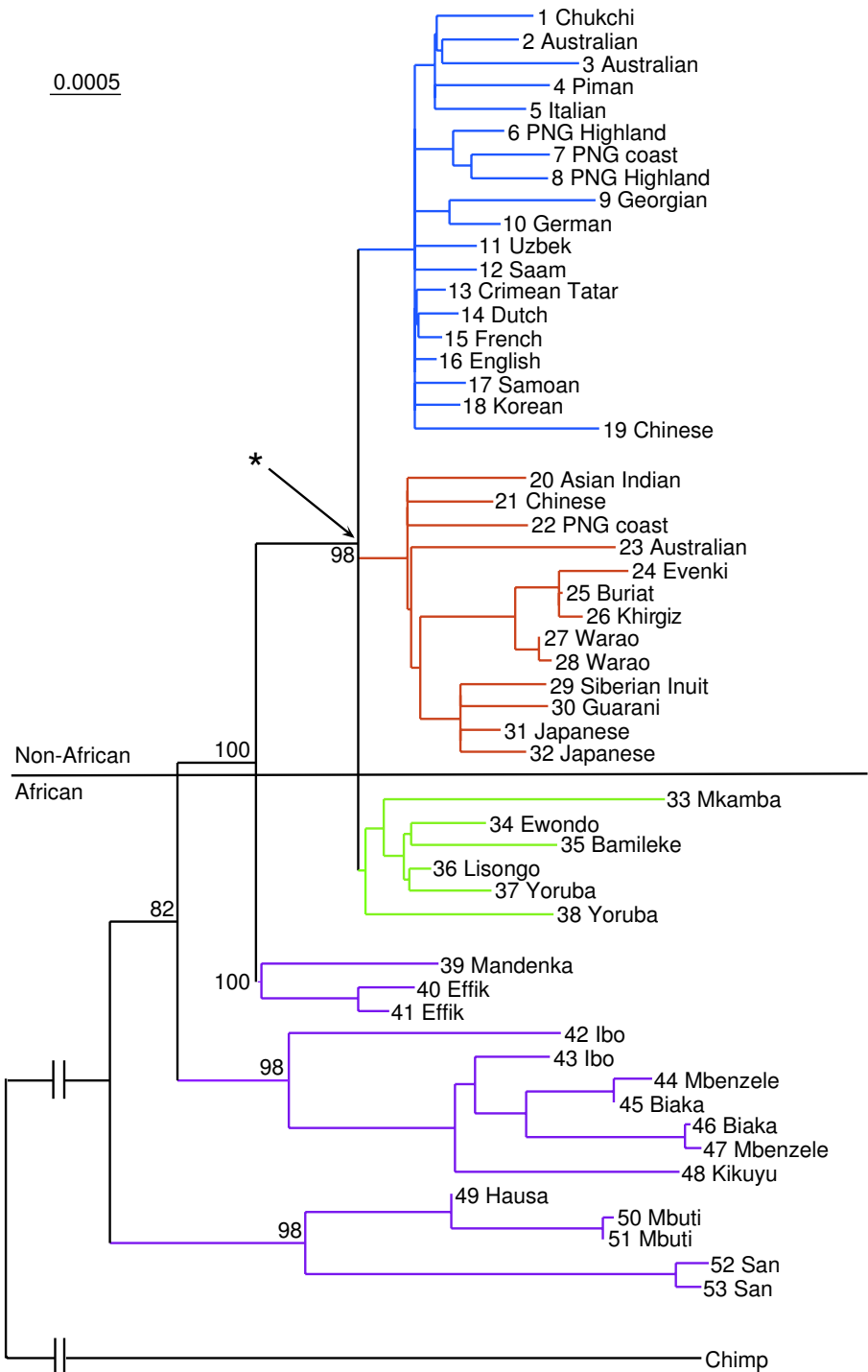
One inferred set of relationships based upon mtDNA sequences is shown in Fig. 12.2. This tree suggests, for example, that native Australians are more closely related to East Asians (e.g., Japanese) than they are to the major groups of Africans. Of particular interest is the observation that populations from the rest of the world are a subset of African populations, supporting the anthropological hypothesis that modern humans in other parts of the world are descendants of migrants that originated from an African population (the “Out-of-Africa” hypothesis). Also noteworthy is the diversity of African populations. For example, the Kikuyu people differ more from the Effik than Europeans differ from Papua New Guinean highlanders.

12.1.3 Reading Trees

In both examples above, we stated conclusions based upon the phylogenetic trees presented in Figs. 12.1 and 12.2. Here we will be more explicit about how to interpret the iconography. We have already had some preparation for this because of the material presented in Chapter 10.

The tree is seen to be a branching structure generated by successive splittings of prior branches. Ideally, the branches are formed by successive bifurcations (splits of one branch into two), but sometimes the branching order of several taxa cannot be determined, and in that case there are multiple branches emanating from a single branch. Eventually, branches end at tips (or “leaves”), and these represent extant taxa (e.g., strains of HIV or contemporary populations of humans). We can trace back along branches leading to two different taxa (taxon A and taxon B, for instance) until these branches join. This junction represents the most recent common ancestor, C. The branch lengths connecting A and C are proportional to the distance between A and C (e.g., using a distance metric defined in Chapter 10). Under ideal circumstances, the distance between A and B is the sum of distances AC and BC. For the examples shown in Fig. 12.1 and Fig. 12.2, distances are depicted as horizontal lines: the vertical lines joining the horizontal ones are just connectors, and the lengths of the vertical connections are not related to distance. A set of all taxa derived from a particular common ancestor is called a **clade**, and the process of branching is sometimes called **cladogenesis**. A tree that depicts the order of branching of taxa without regard to the distances between them is called a **cladogram**.

Fig. 12.2 (opposite page). [This figure also appears in the color insert.] Phylogenetic relationships among representatives of human populations based upon complete mtDNA sequence data (D loop region excluded). The scale is in units of the number of nucleotide differences per site. Reprinted, with permission, from Ingman M et al. (2000) *Nature* 408:708–713. Copyright 2000 Nature Publishing Group.



To construct a tree, we often must identify the ancestral state of the characters employed (e.g., the ancestral DNA sequence). To do this, we employ an **outgroup**, which is a taxon that is clearly more distantly related to the taxa of interest than any of them is to another of these taxa. In Fig. 12.2, the outgroup is chimpanzee. If we look at the branches of humans at the bottom, we see results for two San individuals, preceded above by branches for two Mbuti, and then above those (skipping the Hausa) we see the branch for a Kikuyu individual. Using the convention described above, we can see that the distances separating the two San is less than the distance separating either of the San from either of the Mbuti, indicating (as expected) that the San are more closely related to each other than either is to either of the Mbuti.

The depiction of the data should be distinguished from the quantitative aspects, which relate to branch lengths and branching topology. The top-to-bottom ordering of the branches in this representation is chosen for convenience and has no quantitative significance. If we transposed the top-to-bottom order of the branches leading to the San and the Mbuti+Hausa, the tree would have the same meaning. Similarly, the chimpanzee branch could be “flipped” up to the top, to lie adjacent to the non-Africans, with the same meaning as before. If we look at the lower African branches, we notice that branches emanate from longer branches, whereas non-Africans are related to each other through shorter branches. This implies that African populations diverged from each other earlier, which could produce greater genetic diversity among them. More sampling of African populations is needed to explore this point. Trees help us to interpret the data. For example, notice the pattern of the ten taxa at the top of Fig. 12.1. The first four branches of HIV-1 isolates (HIV-1/U455, /LA1, /EL1, /YBF30) emanate from one side of a bifurcation, the other of which contains SIVcpzUS and SIVcpzCAM3. This set of six viruses represents one of two branches, the other of which contains SIVcpzGAB1. The four HIV-1 strains mentioned are said to form a *sister group* of SIVcpzUS and SIVcpzCAM3. These six together form a sister group of SIVcpzGAB1, and by parsimony type arguments (see below) they share a common ancestor that was an SIVcpz. The appearance of HIV-1 strains among branches that are otherwise SIVcpz suggests that the first four HIV-1 strains mentioned are derived from SIVcpz as a result of a cross-species transfer event. Considering the next connection back, we find two HIV-1 strains forming a sister group with respect to the first seven mentioned above and derived from an ancestor shared by SIVcpzANT. Again, we can argue that this ancestor was an SIVcpz. Notice that there are two clades of HIV-1 within a larger clade that is otherwise SIVcpz: the clade formed from HIV-1 strains U455, LAI, ELI, and YBF30, and the clade formed from strains MVP5180 and ANT70. This suggests that there were at least two interspecies transfer events involved in generating the first ten viruses.

Trees such as those shown in Figs. 12.1 and 12.2 are inferences based upon a particular data set and employing one of several possible methods of tree construction. Such trees may or may not represent the actual phylogeny or

evolutionary tree. There are underlying assumptions used in the construction of any tree, as will become clear in the following sections. It is possible to pour data thoughtlessly into a tree-building software application and get a “result” at the end. If the data are poor or represent inappropriate sampling, or the assumptions for building the tree don’t match the data, then the result may be misleading or worthless.

12.2 Tree Terminology

In the previous section, we discussed two particular examples of trees to illustrate why they are important. In that context, we have already introduced some nomenclature. Now we formalize the presentation. A tree is a particular kind of **graph**, which is defined as a set of vertices connected by **edges**. Each **vertex** has a *degree*, which is the number of edges that emanate from that vertex. A **directed graph** has a defined ordering between vertices connected by one or more edges. A cycle in a graph is a collection v_1, \dots, v_r of vertices with edges between v_1 and v_2 , v_2 and v_3 , \dots , v_{r-1} and v_r , and finally v_r and v_1 . Trees are cycle-free graphs whose vertices correspond to current or ancestral species or populations. Vertices immediately below a vertex v and connected to it by edges are called the **children** of v . Similarly, a vertex u immediately above v and connected to it by an edge is called the **parent** of v .

12.2.1 Conventions

Trees relate species or other biological entities to each other by invoking ancestor-descendant relationships. In this discussion, we consider relationships between species, but remember that the related objects could be gene sequences or populations within species. The observed species (corresponding to the data) appear at the tips of the branches, and these are sometimes called **leaves**. Leaves are vertices of degree 1. Vertices in the tree where leaves or branches join are also called internal **nodes**. For molecular sequences, the actual data correspond to the terminal vertices or leaves. Internal nodes correspond to ancestral species that are not part of the data. Ancestors temporally precede their descendants, and we can sometimes infer likely character states of these ancestral species. However, these states are not usually directly observed.

Binary trees are ones for which each internal node has two children. Internal nodes are connected by internal branches, and leaves are connected to the rest of the tree by external branches emanating from an internal node. The lengths of the branches connecting leaves to nodes and nodes to nodes correspond to distances between them. The trees discussed here are all binary, or bifurcating.

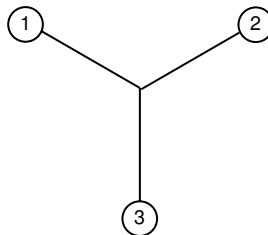
A tree is said to be rooted (or have a **root**) if there is a single ancestral node from which all other nodes descend, and in this case the root node is connected to two branches. Trees may be either rooted or unrooted, as illustrated in Fig. 12.3. If the tree is rooted, direction is defined by the evolutionary time-scale, and we usually associate increasing time with the downward or rightward direction. If there is no root, all we can say is that the leaves correspond to the ultimate descendants of some ancestor, but we do not know whether the internal node adjacent to any leaf is its ancestor, nor do we know the ancestral relationships of the internal nodes. Put another way, we don't know which direction along the edges corresponds to increasing time. In the unrooted tree shown in Fig. 12.3A, OTU 7 may be an ancestor of 8, corresponding to placement of the root as indicated by arrow R1, or 8 might be an ancestor of 7, as would be the case if the root were placed as indicated by arrow R2. Once the root is placed, the ancestor-descendant orientation of all edges is established. Figure 12.3B shows the two different rooted trees corresponding to the two different placements of the root in Figure. 12.3A.

We have so far been representing trees graphically, and we will continue to do so. However, note that there are other representations. For instance, if we construct a tree with branches like those illustrated in Fig. 10.3, that tree could be represented as

$$(((\text{Ho}, \text{Pa}), \text{Go}), \text{Hy}).$$

12.2.2 Numbers of Trees

Before proceeding to a discussion of methods for constructing trees, we need to be aware of the magnitude of the problem. What we seek is the tree or collection of trees that best represents the ancestor-descendant relationships implicit in the data. We might naively imagine that we could draw all possible trees and see how well each fits the data. The unsuitability of this approach is evident if we calculate the possible trees that can be drawn relating n different species. We first consider unrooted trees. The simplest unrooted tree is shown below.



It connects three species, has a single internal node, and contains no internal branches. The number of internal branches is thus $n - 3 = 0$. Now consider the case $n = 4$ by adding one leaf or species (connected between the internal node and 1, for example), as shown in the diagram below.

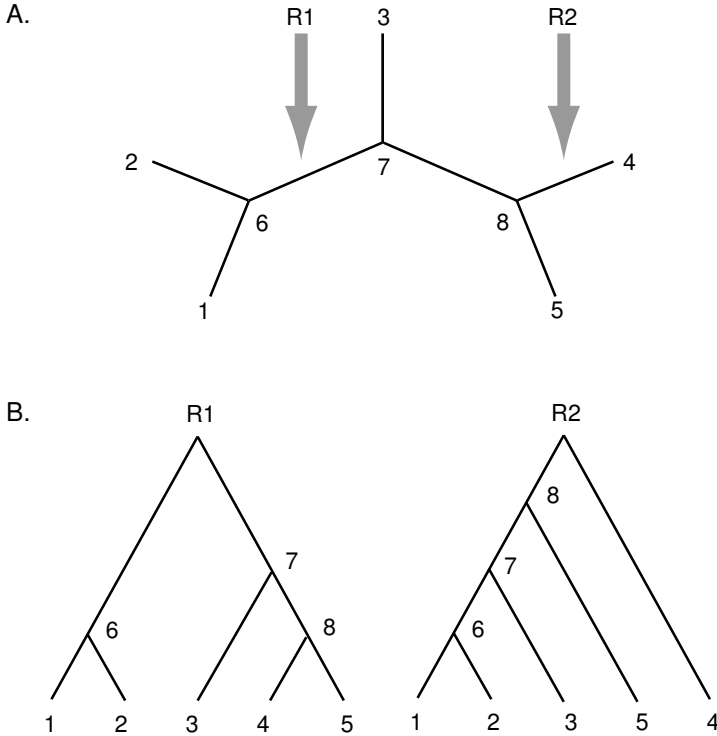
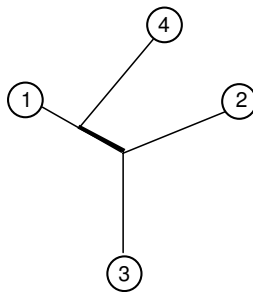


Fig. 12.3. Distinctions between unrooted (panel A) and rooted (panel B) trees. In panel A, R1 and R2 represent two of the seven possible edges in which the root could be placed. Rooted trees corresponding to these two placements are shown in panel B. Note that in the absence of a root, the ancestor-descendant relationships among the vertices are not established beyond the trivial observation that leaves are not ancestors. R1 and R2 imply very different ancestors for most taxa.



There is now one additional leaf and one additional node; there is one internal branch (the thick line). The number of internal branches is now $n - 3 = 4 - 3 = 1$. Repeating this step $n - 4$ additional times results in a tree with n leaves and $n - 4$ additional internal branches for a total of $n - 3$ internal branches.

The total number of branches is therefore the sum of n external branches and $n - 3$ internal branches. There are thus $2n - 3$ branches in an unrooted tree with n leaves.

We now find the number b_n of unrooted trees having n leaves; clearly $b_3 = 1$. We calculate this number by first asking how many ways there are to connect the last leaf; it is just the total number of branches for $n - 1$ leaves. Then we ask how many ways there are to add branch $n - 1$ (which is the number of branches in a tree with $n - 2$ leaves) and so on until we reach $n = 1$. The total number of possible trees is the product of the numbers of ways of adding each successive branch. The number of ways of adding the last branch is

$$b_n = [2(n - 1) - 3]b_{n-1} = (2n - 5)b_{n-1}, \quad n = 3, 4, \dots .$$

The number of ways of adding the second-to-last branch is

$$b_{n-1} = [2(n - 2) - 3]b_{n-2} = (2n - 7)b_{n-2}, \quad n = 4, 5, \dots .$$

The terms may now be calculated successively until reaching $b_3 = 1$. Hence

$$\begin{aligned} b_n &= (2n - 5) \times (2n - 7) \times \dots \times 3 \times 1 \\ &= \frac{(2n - 5)!}{(n - 3)!2^{n-3}}, \quad n = 3, 4, \dots . \end{aligned} \tag{12.1}$$

The proof of the last equality appears in the exercises.

The number of rooted trees with n leaves is closely related to the number of unrooted trees. All that is required is the multiplicative term for the number of ways of placing the root on one of the branches. That number is equal to the number of branches, which we saw above was $2n - 3$. Thus the number of rooted trees b'_n for n leaves is

$$\begin{aligned} b'_n &= (2n - 3)b_n \\ &= \frac{(2n - 3)!}{(n - 2)!2^{n-2}}, \quad n = 3, 4, \dots . \end{aligned}$$

Now we are ready to calculate how many unrooted trees there are for several values of n . The result is shown in Table 12.1. For the example in Fig. 12.1, there were 24 leaves on the unrooted tree. The number of possible trees in that case is then approximately 5.64×10^{26} . With the naive approach, we would have to search through all these trees to find the “best” one; this is currently a computationally impossible task.

12.3 Parsimony and Distance Methods

There are three basic methods for building trees from molecular data: parsimony methods, distance methods, and likelihood-based methods. In this section, we briefly introduce parsimony and distance methods. We defer discussion of the likelihood method to Section 12.5.

Table 12.1. Numbers of possible unrooted trees b_n corresponding to different numbers of taxa or leaves, n .

n	3	4	5	6	7	8	9	10
b_n	1	3	15	105	954	10,395	135,135	2,027,025

12.3.1 Parsimony Methods

Finding a tree relating n species or sequences by parsimony employs the idea that the tree requiring the least number of mutations to relate those sequences is the preferred one. This approach is a specific application of Ockham's Razor: "Plurality should not be posited without necessity." We illustrate this with an example. The method tests hypothetical trees and then calculates the number of base changes needed to yield the observed data.

Suppose the observed data are:

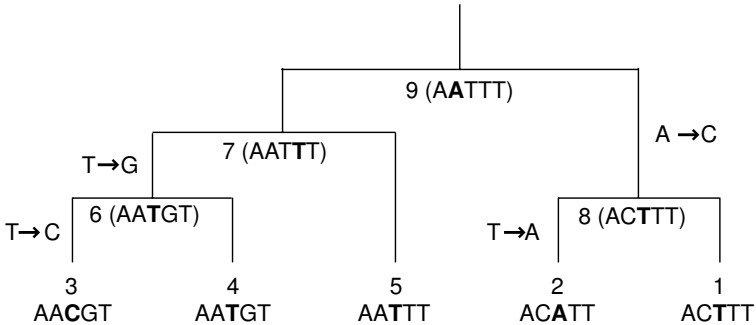
	Site:
	1 2 3 4 5
Species:	
1	A C T T T
2	A C A T T
3	A A C G T
4	A A T G T
5	A A T T T

Expressions in the last section indicate that there are 105 possible rooted trees. We illustrate two of these alternative trees and evaluate which one is most parsimonious. First, we examine the tree $((((3,4),5),1),2)$ with root at 9, diagrammed in Fig. 12.4A. What is the smallest number of changes required to produce the five sequences according to the species relationships implied by this particular tree? We have drawn a set of four mutations that explains the data given this tree. (We illustrate how to obtain these particular mutations a bit later.)

Now let's consider the tree $(((((1,2),3),4),),5)$ diagrammed in Fig. 12.4B. Notice that this tree requires at least six mutations to account for the relationships that the tree implies. Since the second tree requires a greater number of changes than does the first, it is less parsimonious. We therefore prefer the first tree (Fig. 12.4A). Is this tree the correct one? We actually don't know. All we can say is that the probability of mutation at any position is usually very small over relatively short timescales (e.g., 1 to 100 million years) and that the probability of six independent mutations is less than the probability of four mutations.

But how did we decide what the mutations should be and what the sequences at the ancestral nodes might have been? We focus on one position,

A.



B.

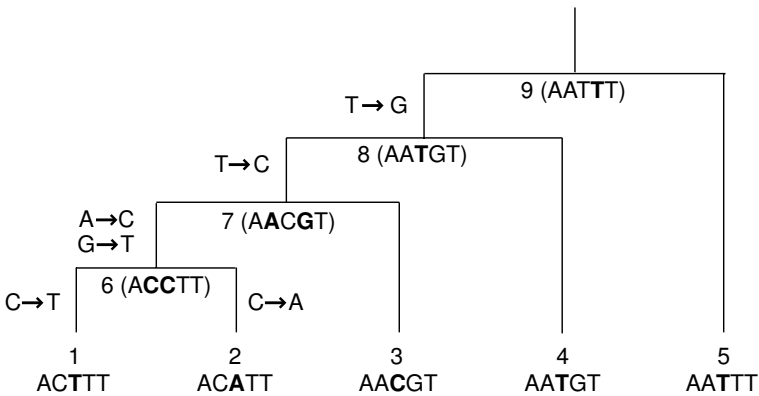


Fig. 12.4. Alternative trees for five different taxa. Nodes are indicated by numbers. The sequences in parentheses at internal and ancestral nodes would normally be inferred. Mutations required to transform the sequence at any particular node to the sequence at the node immediately below are indicated on the appropriate branches. Positions at ancestral nodes altered to produce the mutant descendant are highlighted. The highlighted position at each of the leaves is used for the parsimony illustration in the text. Panel A: More parsimonious tree. Panel B: less parsimonious tree.

position 3, and determine the smallest number of changes needed at that position to be consistent with the tree in Fig. 12.4A. (Bases at position 3 for the five leaves are indicated in boldface type.) Start for example, with the (3, 4) join, corresponding to the letters {C, T}. The parent node, labeled 6, can therefore be either C or T. Next consider joining species 5, which has a T at that site, at the node labeled 7. The most parsimonious assignment is to put T at node 7. If we were to put a C in position 3 at node 7, we would then need to add another mutation leading to 5, which is less parsimonious. Now

consider joining species 1 and 2 at node 8. The sites are A, T, so that at node 8 we must have either A or T. Thus node 8 is {A, T} and node 7 is T. Thus the most parsimonious assignment is to put the T at node 8. This allows us to read off the assignment at each internal node: T (8), T (7), T (6). The resulting assignments of bases at position 3 imply two base changes over the tree. The same method is employed for the five other positions to determine the total number of changes for the tree as a whole.

We now formalize the approach a little. It is convenient to think of the tree as rooted, as for example in Fig. 12.4A. The root node is labeled $2n - 1$, which in Fig. 12.4A is $2 \times 5 - 1 = 9$. We are going to work our way up the tree in the following way. Let F_i denote the set of possible base assignments at a node labeled i , and let L_i denote the number of changes that have accumulated up to that node. We start with $L_i = 0$ if i is a leaf. We want to obtain values (F_3, L_3) at the next node up as a result of combining two nodes with values (F_1, L_2) and (F_2, L_2) . The rules for doing this are as follows:

1. If $F_1 \cap F_2 = \emptyset$, then $L_3 = L_1 + L_2 + 1$, $F_3 = F_1 \cup F_2$.
2. If $F_1 \cap F_2 \neq \emptyset$, then $L_3 = L_1 + L_2$, $F_3 = F_1 \cap F_2$.

Reminder: For sets A and B ,

$A \cap B$, read as “ A intersect B ,” denotes the elements in A and B that appear both in A and in B .

$A \cup B$, read as “ A union B ,” denotes the set of elements that appear in A or B or both.

\emptyset , called the “empty set,” is a set that contains no elements at all.

Let’s try this out at position 4 using the tree in Fig. 12.4A. We start with $(F_3, L_3) = (\{G\}, 0)$ and $(F_4, L_4) = (\{G\}, 0)$, so that $(F_6, L_6) = (\{G\}, 0)$ as a result of applying rule 2. Moving onto node 7, we have $(F_5, L_5) = (\{T\}, 0)$, so that $(F_7, L_7) = (\{G, T\}, 1)$ as a result of applying rule 1 for nodes 6 and 7. Next, since $(F_1, L_1) = (\{T\}, 0)$ and $(F_2, L_2) = (\{T\}, 0)$, we have $(F_8, L_8) = (\{T\}, 0)$ (rule 2). To complete the assignments at node 9, we note that $(F_7, L_7) = (\{G, T\}, 1)$ and $(F_8, L_8) = (\{T\}, 0)$, which by rule 2 yields $(F_9, L_9) = (\{T\}, 1)$. Now that we are at the top, we can read off the number of changes required in the tree from $L_9 = 1$. We have explored all possible paths to the root, and we then backtrack down the tree to construct the assignments to the ancestral states and the base changes that have occurred at each step. At position 4, the change was $T \rightarrow G$ after node 7, leading to node 6.

We can perform this calculation for each site in the sequences related by the tree. The total parsimony cost for a tree is the sum of the parsimony scores for all the sites. The method in its simplest form is to compute the parsimony cost of all possible trees, and choose the minimum-cost tree. Reconstructing the collection of best trees by this approach can be hard though! (The reason is indicated in Section 12.2.2). For larger trees, a variety of heuristic search

methods are used to attempt to identify the best ones without examining all of the trees. There are also versions of weighted parsimony in which different weights are given to different types of substitutions.

12.3.2 Distance Methods

Assume for the moment that we have a set of pairwise distances linking n sequences, and write d_{ij} for the distance between sequence i and sequence j . Let D be the distance matrix whose elements are d_{ij} . This is formally the same type of data employed for the hierarchical clustering described in Chapter 10. In the evolutionary context, the distances meet the three criteria listed in Section 10.3.1. Now that we are interpreting the data in the context of a tree, we need to recognize that biological mechanisms may impose additional conditions on the distances under particular circumstances. These are described below.

We call a tree *additive* if the distance between any pair of leaves is the sum of the distances between those leaves and the first node that they share in the tree. Furthermore, a rooted **additive tree** is called *ultrametric* (or clock-like) if the distances between any two leaves and their common ancestor are equal. An example of an **ultrametric tree** is shown in Fig. 12.5. There is an interesting and important biological concept related to the idea of ultrametric trees. In actuality, mutations may not occur at the same rate in different branches of the tree. For example, mutations accumulate more slowly among primates than among rodents. If the ticking of the molecular clock is not at the same rate on two different branches, then the branch lengths for a given amount of time since the last common ancestor will differ. With ultrametric trees, however, clocks “tick” at the same rate on both branches emanating from a bifurcation.

One natural question is: Given a distance matrix D , can we determine whether we can construct an additive or ultrametric tree corresponding to those distances? The answer is contained in the following result. Let D be the distance matrix.

- (i) Three-point condition for ultrametric trees: D corresponds to an ultrametric tree if and only if for any three sequences i, j, k , the distances satisfy $d_{ij} \leq \max(d_{ik}, d_{kj})$.
- (ii) Four-point condition for additive trees: D corresponds to an additive tree if and only if for any four sequences (labeled here 1, 2, 3, 4) two of the sums $d_{12} + d_{34}, d_{13} + d_{24}, d_{14} + d_{23}$ are equal and greater than or equal to the third.

The interpretation of part (ii) becomes evident when we sketch out trees with three or four leaves and examine the relevant sums (Fig. 12.6). Note that if the tree is ultrametric, part (i) states that two distances must be the same and the third must be smaller. This is shown algebraically and graphically in Figs. 12.6A and 12.6B, respectively.

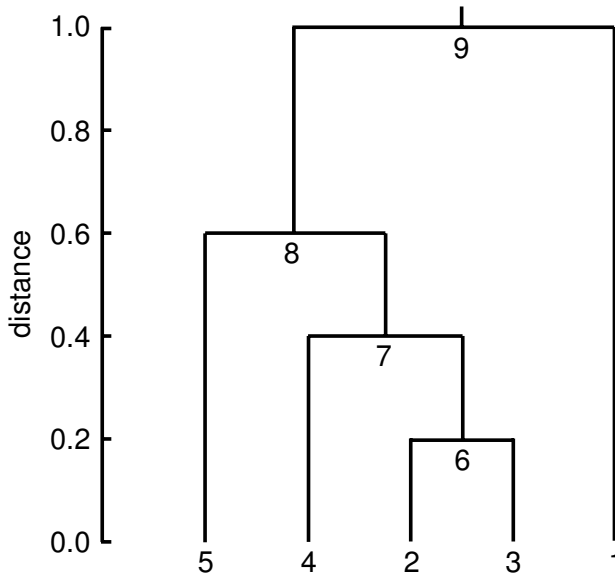


Fig. 12.5. Example of an ultrametric tree. The distance from the leaves to node 6 = 0.2, to node 7 = 0.4, to node 8 = 0.6, and to node 9 = 1. Note the equal lengths of the branches from any node to all leaves derived from that node. This is a distinguishing feature of ultrametric trees.

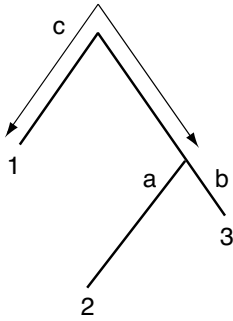
The hierarchical clustering method discussed in Chapter 10 is one way to calculate an ultrametric tree from a set of distances. When the distance between two clusters is defined by group average linkage (see Section 10.4.1), the method produces what is called the UPGMA tree (UPGMA meaning “unweighted pair group method with arithmetic means”). Finding an additive tree is also an interesting problem, but in practice the distances estimated from molecular sequence data do not satisfy the conditions of the theorem and we are left with more work to do. There are several approaches. For an approximately additive tree, a popular method uses Saitou and Nei’s neighbor joining method, implemented for example in the software package MEGA.

Another approach uses weighted least squares to fit a best additive tree for which the distances in any tree to be evaluated are denoted by parameters p_{ij} and the observed pairwise distances d_{ij} as before. We find the assignment of sequences or species to nodes that minimizes the function E given by

$$E = \sum_i \sum_j w_{ij} |d_{ij} - p_{ij}|^\alpha,$$

where α typically has the value 1 or 2 and the w_{ij} are weights. Choices of weights such as $w_{ij} = 1$, d_{ij}^{-1} , or d_{ij}^2 are typical. This also allows for missing data by putting $w_{ij} = 0$ if there is no distance for pair (i, j) .

A.



$$d_{ij} \leq \max(d_{ik}, d_{kj})$$

for all $i, j, k = 1, 2, 3$

$$\Rightarrow \min(a, b) \leq c,$$

$$\min(c, a) \leq b, \text{ and}$$

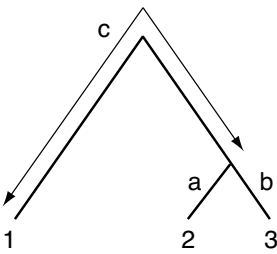
$$\min(c, b) \leq a.$$

If $b \leq a$, we have $b \leq c$,
so $b \leq \min(a, c)$.

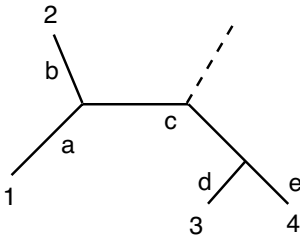
Hence, if $c \geq a$, $b = \min(a, c) = a$.

\therefore the tree is ultrametric
($c \leq a$ is similar).

B.



C.



$$d_{12} + d_{34} = a + b + d + e$$

$$d_{13} + d_{24} = (a + c + d) + (b + c + e)$$

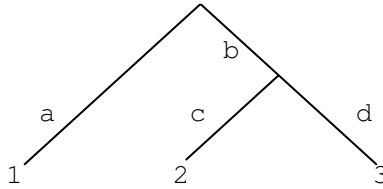
$$= (a + b + d + e) + 2c$$

$$d_{14} + d_{23} = (a + c + e) + (b + c + d)$$

$$= (a + b + d + e) + 2c$$

Fig. 12.6. Illustrations of distance constraints imposed by the ultrametric requirement (panels A and B) and additive distances (panel C). Panel A: An arbitrary relationship between three taxa 1, 2, and 3 is shown with distances from the common node indicated. Assuming the three conditions shows the tree is ultrametric. Panel B: With taxa at the leaves of a rooted tree, the ultrametric condition $(a + b) \leq (a + c) = (b + c)$ can be observed directly. Panel C: Illustration of the four-point condition for additive trees (or a portion of a larger additive tree). The dashed line indicates a potential connection to a larger tree. As indicated earlier, two of the summed distances are equal to each other and larger than the third summed distance.

An example serves to illustrate the rationale behind this approach. Consider the tree for three taxa shown below:



We assign parameters a, b, c , and d to each branch segment, with $a, b, c, d \geq 0$. For this particular tree, $p_{12} = a + b + c$, $p_{13} = a + b + d$, and $p_{23} = c + d$. If we select $\alpha = 2$ and let $w_{ij} = 1$, then according to the equation, we seek to minimize the sum

$$E = [d_{12} - (a + b + c)]^2 + [d_{13} - (a + b + d)]^2 + [d_{23} - (c + d)]^2,$$

with the appropriate choices of a, b, c , and $d \geq 0$. In practice, if the number of sequences or taxa is sufficiently small, one calculates the optimal value of E using numerical analysis methods. The tree having the lowest value of E is the preferred tree.

12.4 Models for Mutations and Estimation of Distances

There are two additional conceptual issues to be addressed before we proceed further. The first concept is modeling the evolutionary process, and the second is using this model to obtain the distance matrix, given that this evolutionary process is operating. In Section 12.1, we indicated that a difference between hierarchical clustering and construction of trees is the inclusion of evolutionary models in the latter case. Before proceeding to the actual construction of trees, we need to introduce a quantitative description of the evolutionary processes that influence the distances used for tree construction. An important aspect of this is the introduction of time and its relationship to distances.

We use a simple stochastic model for the evolution of a DNA sequence through time. Much of the requisite mathematical framework has already been presented in Chapters 2 and 3. For simplicity, we assume that changes in the sequences occur through base substitutions only. Suppose that we have two aligned DNA sequences (in which any sites with gaps have been removed). How can we measure the distance between the two sequences? We have already used the Hamming distance, calculated as the proportion of sites at which the two sequences differ. This measure is not sufficient for many evolutionary studies because it does not allow for the possibility of repeated substitutions at the same site. For example, at a given position in the sequence, there might be an initial $A \rightarrow G$ transition that later reverts by a $G \rightarrow A$ transition, restoring the original state at this position. Because of this phenomenon, we might underestimate the true number of substitutions between our sequences.

We treat the substitution process as a stochastic process and define the distance between the two sequences, measured on the path in a tree separating them, as the expected number of substitutions that change the base at that site. Note that the Hamming distance counts a difference between two sequences at a position as one change. The new definition of distance recognizes that a sequence difference at a position may be the result of more than one change.

12.4.1 A Stochastic Model for Base Substitutions

We consider first a single homologous site in our two sequences and assume the two sites have diverged for time length t . Since the evolutionary clock is running on both paths from the two sequences back to their common ancestor, they are separated by time $2t$. Suppose that the number of substitutions that arise in any branch of length t has a Poisson distribution with mean λt ; the probability that k substitutions arise is given by the Poisson probability

$$\frac{e^{-\lambda t}(\lambda t)^k}{k!}, \quad k = 0, 1, 2, \dots \quad (12.2)$$

We then have to decide what happens at each site where a potential substitution can occur. A general model specifies

$$\mathbb{P}(\text{substitution results in base } j \mid \text{site was base } i) = m_{ij}.$$

A simple example of this mechanism, Felsenstein's (1981) model, specifies

$$m_{ij} = \pi_j, \quad (12.3)$$

with $\pi_i \geq 0$ and $\pi_1 + \pi_2 + \pi_3 + \pi_4 = 1$. From here on we employ the numeric notation introduced previously: 1, 2, 3, and 4 correspond to bases A, G, C, and T, respectively. This implies that the substitution that appears at a position in the sequence does not depend on the type of the base at that position when the substitution occurred. We also assume that the set of probabilities π_j is the same at every position in the sequence.

Next we calculate the probability $q_{ij}(t)$ that a base that was i at time 0 has mutated to base j a time t later. For the model in (12.3), this is easy to do. We calculate the probability by conditioning on whether or not any mutations occurred in time t . Suppose first that $i = j$. If there were no mutations on that branch (probability $e^{-\lambda t}$), then the base time t later must still be j . On the other hand, if there is at least one mutation, then the chance that the resulting base is j is just π_j ; this follows because the mutation mechanism does not care about the type at a site when a mutation occurs. Summing over the two possibilities gives

$$q_{jj}(t) = e^{-\lambda t} + (1 - e^{-\lambda t})\pi_j. \quad (12.4)$$

When $i \neq j$, the same argument shows that

$$q_{ij}(t) = (1 - e^{-\lambda t})\pi_j. \quad (12.5)$$

We need to assume something about the base that is the type of the most recent common ancestor of the two sites we are considering. The usual assumption is to make the evolution of base frequencies along a single branch **stationary**. By stationary we mean that the distribution of base frequencies is the same for every time t . Thus, if π_j^0 denotes the chance that the ancestral base is of type j , then we want

$$\mathbb{P}(\text{base a time } t \text{ later} = j) = \pi_j^0$$

for any time t . Finding the probability distribution that makes this last equation true requires some linear algebra. It turns out that we have to solve the equations

$$\pi_j^0 = \sum_i \pi_i^0 m_{ij} \quad (12.6)$$

subject to $\sum_i \pi_i^0 = 1$. It can be checked that for the model (12.3),

$$\pi_j^0 = \pi_j, \quad j = 1, 2, 3, 4. \quad (12.7)$$

If the π_i^0 do satisfy (12.6), then it can be shown that for any $t > 0$

$$\pi_j^0 = \sum_i \pi_i^0 q_{ij}(t), \quad (12.8)$$

so that, indeed, the base frequencies are not evolving with time.

12.4.2 Estimating Distances

For building the tree, we usually employ distances that take into account the mutation mechanism. Here we present the mathematical background for determining the distance K for any position. The set of K_{ij} for a set of taxa or sequences forms the elements of the distance matrix, analogous to those distance matrix elements employed for clustering in Chapter 10.

Mean Number of Substitutions in Time t

We know that an average of λt substitutions occur at a particular site on a branch of length t . However some of these substitutions result in a given base being “replaced” by the same base. In terms of the end product, this would not be detected as a substitution at all. Now consider a particular substitution. Because of the stationarity assumption, the base at this time is i with chance π_i . In the model in (12.3), the chance that a base of type i changes is just $1 - \pi_i$. Hence the chance that a mutation results in a change of base is

$$H = \sum_{i=1}^4 \pi_i(1 - \pi_i) = 1 - \sum_{i=1}^4 \pi_i^2, \quad (12.9)$$

and the average number of real substitutions in time t is therefore (average number of mutations) \times (proportion that result in change) $= (\lambda t)H$. In the general case with stationary ancestral frequencies, this number can be calculated as

$$\lambda t \sum_i \pi_i (1 - m_{ii}).$$

If we assume the same mutation mechanism applies in the path from the common ancestor to each of its descendants then the expected number of real substitutions between the two present-day bases for the model in (12.3) is

$$K = 2\lambda t H. \quad (12.10)$$

We use K as our measure of distance.

Estimating K

We want to estimate K from the sequence data. To do this, we first have to calculate the chance that we observe an i in one species and a j in the other when they have diverged for time t . We denote this probability by $F_{ij}(t)$. By averaging over the possible ancestral nucleotides, we get

$$F_{ij}(t) = \sum_l \pi_l q_{li}(t) q_{lj}(t), \quad (12.11)$$

assuming that the two sites evolved independently of each other after splitting from their common ancestor.

The result in (12.11) can be simplified if the mutation process is *reversible* with respect to the stationary distribution π . This means that the *detailed balance equations*

$$\pi_i m_{ij} = \pi_j m_{ji} \text{ for all } i \neq j$$

hold. From this it can be shown that

$$\pi_i q_{ij}(t) = \pi_j q_{ji}(t) \text{ for all } i, j \text{ and } t > 0. \quad (12.12)$$

It is easy to check that the model in (12.3) is reversible. When the model is indeed reversible the result in (12.11) can be simplified, since

$$\begin{aligned} F_{ij}(t) &= \sum_l [\pi_l q_{li}(t)] q_{lj}(t) = \sum_l [\pi_i q_{il}(t)] q_{lj}(t) \\ &= \pi_i \sum_l q_{il}(t) q_{lj}(t) = \pi_i q_{ij}(2t). \end{aligned} \quad (12.13)$$

The second equality follows from reversibility and the final equality from elementary probability considerations.

Next we compute the probability $F = F(t)$ that the letters at a particular position in two immediate descendants from the same node are identical. Averaging over the possible ancestral bases gives

$$F = \sum_i \pi_i q_{ii}(2t)$$

for a reversible model, and for the model in (12.3), this reduces to

$$F = e^{-2\lambda t} + (1 - e^{-2\lambda t})(1 - H). \quad (12.14)$$

Putting the Sites Together

We assume that sites evolve independently of one another, with identical mutation processes at each site. We discuss these assumptions later. We can now glue together the information from each site in the sequence by thinking in coin-tossing terms. Look at the aligned sequences (with no gaps), and define

$$X_i = \begin{cases} 1, & \text{if the } i\text{th pair of sites differs,} \\ 0, & \text{otherwise.} \end{cases}$$

If there are s sites in all, then the X_i are independent and (for the model (12.3)) satisfy

$$\mathbb{P}(X_i = 1) = 1 - F = (1 - e^{-2\lambda t})H. \quad (12.15)$$

Then $D = X_1 + \cdots + X_s$, the number of mismatched pairs of bases, is a binomial random variable with parameters s and $1 - F$. (Note that D is the Hamming distance.)

Of course, F is unknown and we have to estimate it from the sequence data. Returning to the coin-tossing example, we know that a sensible estimator of the binomial success probability is the observed proportion of successes. Thus we can estimate $1 - F$ with the quantity D/s . We can then use this to estimate K by solving the equation

$$\frac{D}{s} = (1 - e^{-2\lambda t})H,$$

obtaining (after taking logs and simplifying)

$$2\lambda t = -\log\left(1 - \frac{D}{sH}\right).$$

Recalling the definition of K in (12.10), we obtain

$$K = 2\lambda tH = -H \log\left(1 - \frac{D}{sH}\right). \quad (12.16)$$

This estimate of the distance K between the two sequences is known as the *Jukes-Cantor formula*.

If we knew H , we could estimate K using the Jukes-Cantor formula. Typically we do not know H , and we have to estimate this from the data also. The usual strategy is to estimate the base frequencies $\pi_A, \pi_C, \pi_G, \pi_T$ from

the collection of homologous sequences being compared, and then calculate $H = 1 - \sum_i \pi_i^2$.

Pairwise distances such as K can be calculated for many other models of the evolutionary process. These often arise by changing the form of the mutation parameters m_{ij} in (12.3), and finding the transition probabilities $q_{ij}(t)$ for the new model. The subsequent analysis is simplified if the model is reversible.

12.5 Maximum Likelihood Methods

Now that we have seen how to calculate distances and to build trees based upon those distances, we move to the third of our methods, developing a statistical method for tree-building based upon a maximum likelihood approach. For illustration, we assume that we are trying to construct an ultrametric, or clock-like, tree.

12.5.1 Representing a Tree

A clock-like tree can be represented in many ways. Here is a simple one. First, we label the n species $1, 2, \dots, n$ at the leaves and assign labels $n+1, \dots, 2n-1$ to the $n-2$ internal nodes. This should be done in such a way that any internal node has a label larger than any of its descendants. This labeling lists the names of the descendants of each internal node and hence the complete tree shape. The other missing ingredient is the times of the joins in the tree. For an ultrametric tree, it is natural to record the distance to each internal node measured from the present time back into the past. For example, the tree described by

6	2	3	0.2
7	6	4	0.4
8	7	5	0.6
9	1	8	1.0

has $n = 5$ species. The left-hand column gives the *internal node labels*; these are followed by the labels of the two *direct descendants* and the *distance* from the internal node to the leaves. In the example, species 2 and 3 join first at node 6, 0.2 units back in time; node 6 is then joined to leaf 4 at node 7, the distance being 0.4 units. Leaf 5 joins the tree at node 8, a distance of 0.6 units from the present, and finally leaf 1 joins at the root node 9, a distance of 1 unit from the present. The resulting tree was shown in Fig. 12.5. Note that it is easy to represent other tree topologies using this data structure.

12.5.2 Computing Probabilities on a Tree

We use the evolutionary model of substitutions described in Section 12.3. As earlier, we assume that the sequences under comparison have been aligned

and that any gaps have been removed. We also assume we know the topology of the phylogenetic tree linking the species. First, we calculate the probability $p(i_1, i_2, \dots, i_n)$ that species $1, 2, \dots, n$ have bases i_1, i_2, \dots, i_n at a given nucleotide site. To see what is involved, let's do the calculation for the case of three species with a tree of the form

$$\begin{array}{c} 4 \ 1 \ 2 \ t_1 \\ 5 \ 3 \ 4 \ t_2 \end{array}$$

We get

$$p(i_1, i_2, i_3) = \sum_a \sum_b \pi_a q_{ai_3}(t_2) q_{ab}(t_2 - t_1) q_{bi_2}(t_1) q_{bi_1}(t_1). \quad (12.17)$$

This formula arises by considering the possible assignments a for the root node 5 and b for the node 4 and summing over the possibilities for a and b . Here the π_a are the stationary base frequencies (i.e., base frequencies assuming the same distribution both for ancestral and descendant bases). The $q_{ij}(t)$ give the probability of base j at a site given base i at that site a time t earlier. A formula of this type can be written down for any tree topology, but it can be very complicated.

The standard way to arrange the computations is to use the peeling algorithm, due in this context to Felsenstein (1981). This recursive algorithm goes through the tree calculating at each internal node the probability of the subtree data below that node as a function of the base at the node. This algorithm is routinely implemented in most tree-building software. The principle behind it is to “move all summation signs as far to the right as possible.” See Exercises 11 and 12 for further details.

12.5.3 Maximum Likelihood Estimation

We compute the likelihood of the data by multiplying the likelihood terms $p(i_1, i_2, \dots, i_n)$ calculated using the peeling algorithm at each site in the data. This comes about because we assume that the sites are evolving independently of each other. The maximum likelihood estimator of the parameters of the mutation model and the branch lengths are found in two steps: first, maximize the likelihood over the model parameters for a given tree, and then repeat this maximization over all possible trees. The maximum likelihood estimator we are looking for is the one that maximizes these maximized likelihoods. This recipe is manageable for small numbers of species but impractical for large numbers (bigger than 5, say). In such cases, a variety of heuristic search algorithms have been devised to explore tree space (see, e.g., Hillis et al., 1996).

An important question to be addressed is: What can be estimated? It is conventional to reduce the number of parameters to be estimated by setting the stationary frequencies to be the observed proportions of each base in the entire data set. We do this, too. For a given topology, for the simple mutation

model we are using, there is one parameter, λ , for the substitution rate, as well as $n - 1$ node heights that might be estimated. However, looking at the form of the $q_{ij}(t)$ in (12.4) and (12.5) shows that times and the rate λ are *confounded*; that is, they cannot all be estimated separately. We then have two different (but equivalent) choices:

- (i) We can fix the height of the tree (the distance to the root node) and estimate λ and $n - 2$ other node heights, or
- (ii) We can fix parameter λ (at value 1, say) and estimate the $n - 1$ node heights.

The program used in the exercises allows for either possibility. This observation shows why we need external calibration of the tree height in order to obtain node heights in years and mutation rates in units of changes per year.

12.5.4 Statistics and Trees

Often we want to use molecular data to test different hypotheses about the phylogenetic relationships between species or to estimate (and perhaps find confidence intervals for) parameters in the mutation model (see, e.g., Huelsenbeck and Rannala, 1997). A common guiding heuristic is to compare likelihoods among different models. This is a subtle business, but a couple of examples should illustrate what can be done.

One problem concerns whether or not a given phylogeny explains a data set better than another phylogeny. We have already discussed this problem in our description of parsimony and distance methods. In the present context, we could compare the log-likelihoods of the data under each model; presumably the one with the highest log-likelihood provides a better description of the data. This can be formalised into a rigorous statistical procedure (e.g., Goldman, 1993), but we do not present the details here.

Another problem concerns estimation of parameters in a mutation model given a particular phylogenetic tree topology. We can test (formally at least) whether a simpler model (the “reduced hypothesis” in what follows) provides an adequate fit to a data set, as opposed to a model with more parameters (the “general hypothesis”). We can use a χ^2 test as follows:

1. Calculate the maximized likelihood L_1 under the general hypothesis.
2. Calculate the corresponding likelihood L_0 under the reduced hypothesis.
3. Calculate $W = -2[\log L_0 - \log L_1]$.
4. Under the reduced hypothesis, the distribution of W is approximately χ^2 with degrees of freedom given by the difference between the dimension of the general hypothesis and the dimension of the reduced one.

12.6 Problems with Tree-Building

There are numerous problems associated with tree-building using molecular data. We might wonder about the adequacy of a given mutation model as a description of the data. We have made a number of assumptions, among them

- (a) Sites evolve independently of one another.
- (b) Sites evolve according to the same stochastic model.
- (c) The tree is rooted.
- (d) The sequences are aligned.

Point (c) is easy to deal with; there are likelihood methods for unrooted trees (assuming reversible models of substitutions) with the additive property (see, e.g., Felsenstein, 2004). Point (b) is clearly violated in a coding region: it is well-known that third positions evolve faster than first positions, which in turn evolve faster than second positions. It is possible to fit models with a common tree while allowing for this type of rate variation. It is sometimes useful to try a model of random rate variation, in which each site evolves independently but with a rate drawn independently across sites from a given distribution. Some of the computational and biological problems are described in Yang (1996). There have been some attempts to alleviate the difficulties of assumption (a), which can arise, for example, in sequences with secondary structure. Point (d) has proved more difficult to address. Usually, gaps in aligned sequences are removed before tree-building. The effects of this are hard to assess, especially in distantly related species in which alignment is hard. Finally, we remark that there are computational problems with exploring tree space adequately. One way to address this issue is through the use of Bayesian computation, in particular the Markov chain Monte Carlo methods (see Huelsenbeck et al., 2001).

When the goal is to understand the evolutionary relationships between organisms, the distinction between a gene tree and a species tree becomes important. This distinction is illustrated in Fig. 12.7. A **gene tree** is a tree drawn from DNA or protein sequences corresponding to a particular gene shared by a set of organisms. Each different gene may (or may not) produce a different tree for the same set of organisms. A **species tree** is often produced from sets of macroscopic characters but also may be produced from sequence data. Generating a consensus species tree from a collection of gene trees is a difficult problem that is beyond the scope of this book.

References

- Felsenstein J (1981) Evolutionary trees from DNA sequence data: A maximum likelihood approach. *Journal of Molecular Evolution* 17:368–376.
 Felsenstein J (2004) *Inferring Phylogenies*. Sunderland, MA: Sinauer.

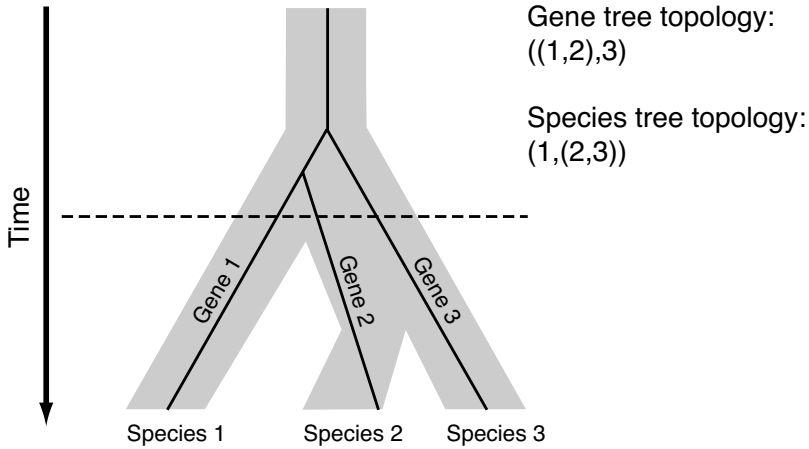


Fig. 12.7. Distinction between gene trees and species trees. The thick black lines denote the gene tree constructed from homologous regions of the same gene found in species 1, 2, and 3. The shaded bands represent the species tree. Underlying this diagram is the concept that species are represented by populations, which at any time may have multiple alleles for any gene. Genes transmitted to the next generation are sampled from these populations. For example, at the time corresponding to the dashed line, there are three alleles of the gene present in a single species. The allele for gene 2 that became fixed in species 2 actually diverged later than the allele that became fixed in species 3. Mechanisms of sampling of alleles from populations are discussed in Chapter 13.

Goldman N (1993) Statistical tests of models of DNA substitution. *Journal of Molecular Evolution* 36:182–198.

Hahn BH, Shaw GM, De Cock KM, Sharp PM (2000) AIDS as a zoonosis: Scientific and public health implications. *Science* 287:607–614.

Hillis DM, Moritz C, Mable BK (1996) *Molecular Systematics* (2nd edition). Sunderland, MA: Sinauer.

Huelsenbeck JP, Rannala B (1997) Phylogenetic methods come of age: Testing hypotheses in an evolutionary context. *Science* 276:227–232.

Huelsenbeck JP, Ronquist F, Nielsen R, Bollback JP (2001) Bayesian inference of phylogeny and its impact on evolutionary biology. *Science* 294:2310–2314.

Ingman M, Kaessmann H, Pääbo S, Gyllenstein U (2000) Mitochondrial genome variation and the origin of modern humans. *Nature* 408:708–713.

Yang Z (1996) Among site rate variation and its impact on phylogenetic analyses. *Trends in Ecology and Evolution* 11:367–372.

Freely available software programs:

<http://evolution.genetics.washington.edu/phylip/software.html>

Site for PHYLIP, maintained by Joe Felsenstein.

<http://www.megasoftware.net>

Site for MEGA2.1, which focuses on distance methods.

<http://abacus.gene.ucl.ac.uk/software/paml.html>

Site for PAML, which focuses on likelihood methods.

<http://morphbank.ebc.uu.se/mrbayes/>

Site for MrBayes. This is good for a Bayesian perspective.

Exercises

Exercise 1. Draw all rooted and unrooted trees with $n = 5$ leaves labelled $\{a, b, c, d, e\}$.

Exercise 2. Use Stirling's approximation

$$x! \sim (2\pi)^{1/2} x^{(x+1/2)} e^{-x}$$

as in Section 6.5 to find the corresponding approximation for b_n in (12.1). Use this to find an approximation for b_{100} .

Exercise 3. Suppose d_{ab} , d_{ac} and d_{bc} are distances that form an additive tree for the unrooted tree with leaves $\{a, b, c\}$. (There is only one such tree.) The tree has length x for the leaf a , length y for the leaf b and length z for the leaf c . Derive a formula for x , y and z in terms of d_{ab} , d_{ac} and d_{bc} . [Hint: Draw a picture.]

Exercise 4. Using the results of the previous problem, find the unique tree with distances given by

	a	b	c	d
a	0	3	6	5
b		0	7	6
c			0	3
d				0

Exercise 5. You are given the following set of species and aligned sequences:

	Site:
	1 2 3 4
Species:	
1	T C A A
2	G C A T
3	T T T T
4	G A T A
5	G A A C
6	A T A G

Find the parsimony score for the tree $((((1, 2), (3, 4)), 5), 6)$. Indicate the F sequence at each vertex of the tree.

Exercise 6. Table 10.1 gives a 60 bp region of the cytochrome oxidase subunit II coding sequence for five primates.

- Write an R function to calculate the matrix of pairwise distances defined in (12.16) for these data.
- Find a UPGMA tree based on these distances, and compare to the results in Section 10.4.

Exercise 7. Discuss the logical connection between the data structure introduced in Section 11.5.1 and the procedure for agglomerative clustering described in Section 10.4.1.

Exercise 8. For the data structure introduced in Section 11.5.1, devise an algorithm that computes the number of leaves below each internal node of a rooted binary tree. Can you modify your algorithm to return the number of nodes and leaves below each internal node?

Exercise 9. Devise an R function that takes a tree described by the data structure in Section 11.5.1 as input, and produces a plot such as the one shown in Fig. 12.5 as output, using the standard R function `plclust` (see Computational Example 10.1).

Exercise 10. Consider a tree with just two leaves. Suppose the branch to species 1 has length t_1 and the branch to species 2 has length t_2 . Show that for a reversible mutation model, the probability F_{ij} of observing base i in species 1 and base j in species 2 depends on t_1 and t_2 only through the sum $t_1 + t_2$. This result is known as the pulley principle, which says that we cannot tell the direction of time. (For the meaning of “pulley principle,” imagine that two leaves are connected by a rope through a pulley mounted at the node immediately above it. If you lengthen the rope on one side of the pulley, the rope length on the other side must correspondingly shorten.)

Exercise 11. This exercise illustrates an important principle in calculating likelihoods on trees. Equation (12.17) gives the probability of observing particular bases at the leaves of a tree with $n = 3$ species as

$$p(i_1, i_2, i_3) = \sum_a \sum_b \pi_a q_{ai_3}(t_2) q_{ab}(t_2 - t_1) q_{bi_2}(t_1) q_{bi_1}(t_1).$$

This can also be written in the form

$$p(i_1, i_2, i_3) = \sum_a \pi_a q_{ai_3}(t_2) \sum_b q_{ab}(t_2 - t_1) q_{bi_2}(t_1) q_{bi_1}(t_1).$$

Evaluate carefully how many addition and multiplication operations are performed in these two formulae, and deduce that the first form is less efficient than the second.

Exercise 12. This exercise, an extension of Exercise 11, provides more details about the peeling algorithm used to compute probabilities on trees. Consider the tree with four species and topology given by $((1, 2), 3), 4)$. Let c denote the internal node linking species 1 and 2, b denote the node linking c and 3, and a denote the root node.

- a. Write a formula for the probability p of observing particular bases at the leaves of the tree in the form

$$\sum_a \sum_b \sum_c \{\text{an expression involving the product of 7 terms}\}$$

and find this expression.

- b. Evaluate the number of addition and multiplication operations needed to evaluate p .
- c. The peeling algorithm is based on the principle of “moving all summation signs as far to the right as possible.” Write down the corresponding formula for p and evaluate the number of addition and multiplication operations needed to calculate it. Comment.
- d. Describe in a few sentences how the peeling algorithm works.

Exercise 13. For this example you need the primate data for *Pongo*, *Pan*, *Gorilla* and *Homo* from Table 10.1. *Pongo* is taken to be the outgroup in this example. This exercise investigates whether *Pan* is more closely related to *Homo* than *Gorilla* is.

- a. Using the approach in Section 12.5.1, give the representation of the tree in which *Pan* and *Homo* split first. Calculate the likelihood of the data for this topology.
- b. Repeat the previous problem, but for the tree in which *Gorilla* and *Homo* split first.
- c. What do you conclude about the divergence of these three species?
- d. What is the effect on the estimated topology of the tree if you use more of the cytochrome oxidase sub-unit sequence? If you use other sequences?

[Note: this exercise can be completed using likelihood software available at www.cmb.usc.edu.]

Genetic Variation in Populations

13.1 The Biological Problem

In Chapter 12, we showed how evolutionary relationships between organisms (taxa or OTUs) could be inferred from DNA sequence data. In that discussion, it was assumed that the variation *among* taxa being analyzed was much greater than any variation *within* taxa, so that each taxon (species) could be represented by a unique set of characters. Because mutations (variations) accumulate over time, this is equivalent to stating that the phylogenetic trees that were constructed corresponded to taxa that had diverged from each other relatively long ago. Suppose, however, that we look at a population of organisms alive today and examine the allele frequencies within this population. What inferences can we make about the history of this population? Now that we are focusing on variation *within* a taxon, we are concerned with a time-scale that is short compared with the time-scale associated with phylogenetic trees.

The concept of population is extremely important for understanding biological organisms and their evolution. A **population** is a localized collection of individuals of a species that are capable of exchanging the genes that characterize that species. The San people of southern Africa or the brown pelicans living on Anacapa Island off the coast of Southern California are examples of populations. A biological population observed today is a “snapshot” within a particular taxon of the current state of a process that has been occurring for about 3.9 billion years. The concept of population is intimately connected with evolution. Evolution may be parsimoniously defined as the process of change over time of the allele frequencies within populations.

In this chapter, we consider the parameters and statistics required to describe the genetic properties and genetic changes in populations of diploid sexual organisms. The criteria used to characterize a population must take into account the dynamic properties of that population. Populations are undergoing constant change because the individuals that comprise the population are themselves changing from birth to mate selection, reproduction, and

eventually death. In addition, individuals may migrate into or away from a particular population. When a portion of a population relocates, the members of that subpopulation introduce a sample of the alleles from the original population into the new locale. The subpopulation may then grow to become different from the original population, possibly leading to a speciation event. When individuals migrate into a population from another distinct population, they may introduce new alleles. Added to these processes of change is the inexorable accumulation of point mutations, which adds new alleles to any population (other types of mutations, such as deletion or inversion, usually inactivate genes rather than create new alleles). It is evident that populations and the forces acting upon them can be complex, thus requiring correspondingly complex statistical models.

There are a number of reasons why we wish to study populations. Arguably the most important is that variation within populations is the basis for natural selection. Another reason is to infer population history. For example, we can measure allele frequencies in human populations and use them to infer migration patterns that affected recent human evolution. In addition, we might look for alleles associated with genes implicated in particular genetic diseases. A special case of this is analyzing genes associated with diseases for which particular human subpopulations have elevated risk (e.g., adult diabetes in some native American populations in the southwestern United States).

13.2 Mendelian Concepts

In Chapter 1, we briefly discussed genetics, deferring detailed discussions to the occasions when they would be needed. Since we describe populations in terms of their genes, it is appropriate at this point to review some basic Mendelian genetics.

Every diploid organism has two copies of each genetic locus carried on pairs of autosomes (chromosomes other than sex chromosomes). A **locus** is an identifiable region on a chromosome, and it may correspond to a gene or to a physical marker such as a sequence-tagged site (STS). We discuss genes for simplicity. Gene nomenclature can vary from organism to organism, and we use conventions like those used in human genetics. *Genes* are represented as uppercase italicized letters (up to four for humans), and *phenotypes* are represented by the corresponding Roman letters: phenotype A corresponds to gene *A*. We are concerned with diploid organisms, and the alleles residing on different members of a corresponding pair of chromosomes are separated by a forward stroke: A_1/A_2 means that allele A_1 of gene *A* resides on one chromosome and allele A_2 resides on the other. The two gene copies corresponding to a particular locus in an organism may or may not be exactly identical. For example, in human populations from equatorial Africa, the β -globin gene *HBB* may appear in at least two forms: type A, which is associated with the normal phenotype, and type S, which is associated with sickle-cell dis-

ease. These alternative forms of the same gene are examples of alleles. During meiosis (see Section 1.3.1), alleles corresponding to a particular locus segregate, which means that one copy of any locus appears in any given gamete. In contrast, two different genes on the same chromosome do not segregate unless recombination has occurred.

If two alleles at a given locus are identical in an individual, then that individual is said to be **homozygous** for the genes at that locus. If the two alleles are different, then the individual is **heterozygous** with respect to the genes at that locus. A special case of homozygosity is **autozygosity**, which means that two alleles are identical because they have descended from the same common ancestral allele. Suppose that phenotype A is associated with allele A_1 , and that phenotype A^- is associated with allele A_2 . If an A_1/A_2 heterozygote has phenotype A, then A_1 is said to be the **dominant** allele and A_2 is said to be the **recessive** allele. In humans, cystic fibrosis is associated with a mutant form of the *CFTR* gene (located on chromosome 7). Individuals who have two copies of the normal gene, or one copy of the normal allele and one copy of the defective allele, are not affected, but persons who have two defective alleles are affected. The wild-type allele is thus dominant, and the disease is said to be autosomal recessive. Sometimes the phenotype of the heterozygote is intermediate between those of the parents, each of whom is homozygous for different alleles. This circumstance is called incomplete dominance. Sometimes the phenotype associated with a gene fails to appear because of the particular constellation of other genes in that individual or particular environmental circumstances. The probability that a gene confers the phenotype associated with it is called its **penetrance**.

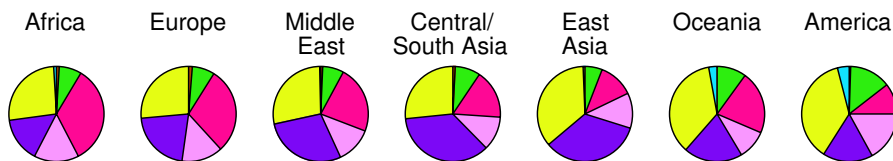
It is very important to note that a large proportion of phenotypes in complex organisms depend upon the cumulative effects of several genes. Such phenotypes are called **polygenic traits**. For example, height and skin color in humans do not depend on single genes but rather on the collaboration of a number of genes. Skin color in humans is thought to be controlled by three to six loci. Teasing out the contributions of multiple genes, each of which might contribute in a quantitative way (so-called quantitative trait loci), can be a complicated exercise in genetics.

13.3 Variation in Human Populations

In this section, we use populations drawn from different regions of the world to show how genetic variation is distributed across human populations. Because only a small proportion of alleles are unique to a particular population, knowing what population or region an individual belongs to does not help much in predicting which alleles the individual will have. On the other hand, by studying information across different loci, we can often predict which population or region an individual comes from by knowing which combination of alleles is present in the individual.

The data are taken from 52 populations in seven geographic regions of the world measured for genotypes at 377 loci (Rosenberg et al., 2002). An example of a region is sub-Saharan Africa, and examples of populations within this region are Bantu, Mandenka, Yoruba, San, Mbuti Pygmy, and Biaka Pygmy. The loci examined were *microsatellites*, which are tandem repeats of short k -words (see Section 13.4.2). Corresponding to each locus is a characteristic number of different alleles. An example of the allele frequencies for two loci measured for seven different geographical regions is shown in Fig. 13.1. For the particular locus D12S2070, there are eight alleles. Clearly, the allele frequencies differ for different regions. For example, one allele represents 85% of the total in native American populations, but that same allele represents only 7% of the total in populations from the Middle East. Allele frequencies for other loci may not vary much across regions, as illustrated by D6S474.

D6S474



D12S2070

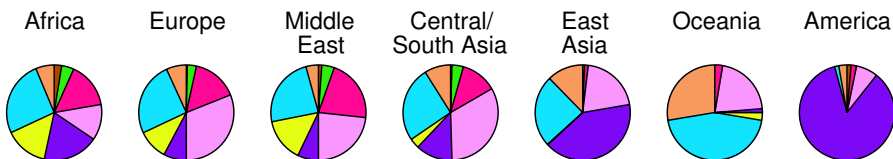


Fig. 13.1. [This figure also appears in the color insert.] Allele frequencies associated with two microsatellite markers in human populations taken from seven different regions. Each allele has a different color code, and the size of the sector in the pie chart indicates allele frequency. Reproduced with permission. Copyright 2002 NA Rosenberg.

13.3.1 Describing Variation Across Populations

We now introduce some statistics for describing population variation. Let K be the number of alleles for any given locus, and let p_j be the relative frequency of allele j . The probability F that two randomly chosen genes at a locus are identical is

$$F = \sum_{i=1}^K p_i^2. \quad (13.1)$$

We note that the population frequencies p_j are unknown parameters of our population and must be estimated from a sample of individuals from that population. F can then be calculated from the estimated allele frequencies. The **heterozygosity** H for any locus is defined as $H = 1 - F$:

$$H = 1 - \sum_{i=1}^K p_i^2. \quad (13.2)$$

Computational Example 13.1: Calculating heterozygosities

Heterozygosity provides a measure of the amount of variation within a population. Exercise 1 at the end of this chapter should now be completed. The results illustrate how H behaves for different numbers of alleles and allele probability distributions. Observe that for a fixed number of alleles, the heterozygosity is greater if the distribution of the allele probabilities is uniform than if some alleles predominate. Also, for uniform allele distributions, the heterozygosity is greater if the number of alleles is greater.

Heterozygosities can be calculated within populations, within regions, or globally. Table 13.1 exemplifies the typical structure of the underlying data for a single locus having three alleles. There would be similar tables for every other locus, each with its characteristic number of alleles. The heterozygosity reported for each population measured over 100 loci would be the heterozygosity averaged over all 100 loci.

Examples of Calculated Heterozygosities

For the data in Table 13.1, we can calculate a variety of heterozygosities.

Population 1, region 1:

$$H_{Pop1-1} = 1 - [(3/10)^2 + (5/10)^2 + (2/10)^2] = 0.620.$$

Region 1 as a whole:

$$H_{R_1} = 1 - [(15/30)^2 + (10/30)^2 + (3/30)^2] = 0.611.$$

Average of populations in region 1 (see (13.3)):

$$H_{R_1,P} = (0.62 + 0.62 + 0.46)/3 = 0.567.$$

World as a whole:

$$H_W = 1 - [(31/120)^2 + (44/120)^2 + (45/120)^2] = 0.658.$$

Table 13.1. Example of data structure and heterozygosities for a single gene having three alleles examined in four world regions, each containing three populations. The three rightmost columns show the number of instances of the alleles a, b, c .

		Allele		
		a	b	c
Region 1	Pop 1-1	3	5	2
	Pop 1-2	5	3	2
	Pop 1-3	7	2	1
	Total for region	15	10	5
Region 2	Pop 2-1	3	3	4
	Pop 2-2	4	3	3
	Pop 2-3	3	4	3
	Total for region	10	10	10
Region 3	Pop 3-1	0	1	9
	Pop 3-2	1	1	8
	Pop 3-3	0	2	8
	Total for region	1	4	25
Region 4	Pop 4-1	2	7	1
	Pop 4-2	1	7	2
	Pop 4-3	2	6	2
	Total for region	5	20	5
World		31	44	45

In addition to the direct measures of heterozygosity, it is useful to consider particular averages, as illustrated in Computational Example 13.1. If there are B populations in a region with heterozygosities H_1, H_2, \dots, H_B , then the average heterozygosity for that region is

$$H_{R,\text{avg}} = \frac{1}{B} \sum_{i=1}^B H_i. \quad (13.3)$$

We can also calculate the average regional heterozygosity over R regions,

$$H_R = \frac{1}{R} \sum_{i=1}^R H_{R_i}, \quad (13.4)$$

where the H_{R_i} correspond to the heterozygosities of each region as a whole, obtained by pooling data from all populations (i.e., not calculated as in (13.3)).

Now we are prepared to take a look at the actual population heterozygosities estimated for different human populations (Table 13.2). The first column

of heterozygosities are the regional data calculated directly from the allele frequencies of the pooled populations—not calculated as averages. The bottom line, first column, is computed from a pool of all the data, taking the world as a single region. Data above the last entry in the second column ($H_{R,\text{avg}}$) were calculated as averages of population heterozygosities using (13.3). The average regional heterozygosity, H_R , calculated according to (13.4), is 0.733. The average subpopulation heterozygosity, H_S (bottom line, second column), can be calculated from the entries above it as a weighted average using the numbers of each population as weighting factors.

Table 13.2. Heterozygosities for 377 human microsatellite alleles by population and region. H_R : region as whole; $H_{R,\text{avg}}$: average of populations in region. The number of populations in each region is given in parentheses. Source: Rosenberg NA et al. (2002) www.sciencemag.org/cgi/content/full/298/5602/2381/DC1.

Region	H_R	$H_{R,\text{avg}}$
sub-Saharan Africa	0.792	0.774 (6)
Europe	0.753	0.751 (8)
Middle East	0.761	0.756 (4)
Central/S. Asia	0.759	0.752 (9)
East Asia	0.730	0.723 (18)
Oceania	0.695	0.683 (2)
America	0.644	0.599 (5)
World	0.771	0.727

Remember that the heterozygosities provide measures of population variation. With the data in Table 13.2 and the population averages described above, we can estimate the fractions of human variation that occur within populations, among populations within regions, and among regions of the world. Let H_T be the heterozygosity for the total population of the world (see Table 13.2, column 1, last entry). Since we can write

$$H_T = H_T - H_R + H_R - H_S + H_S,$$

we see that

$$1 = \frac{(H_T - H_R)}{H_T} + \frac{(H_R - H_S)}{H_T} + \frac{H_S}{H_T}. \quad (13.5)$$

The first term on the right-hand side of (13.5), the fraction of variation that occurs among regions of the world, is $(0.771 - 0.733)/0.771 = 0.049$. The second term, the fraction of variation that occurs among populations within a region, is $(0.733 - 0.727)/0.771 = 0.008$. The last term, the fraction of variation within populations, is $0.727/0.771 = 0.943$. This means that 94% of the human variation in the world occurs at the level of local populations,

and only about 5% of the variation occurs among regions of the world. On average, for each locus, there is more variation between two African people than there is between the average African and the average Chinese person.

The last statement does not mean that we can't distinguish East Asians from Africans. A simple example that shows why can be illustrated from the allele frequency data in Table 13.3. Suppose that we have two populations, I and II, and estimated allele frequencies at three unlinked loci, each of which has two alleles. We score a particular individual (Joe) for his genotype with respect to these three genes and find him to be A_1/A_1 , B_1/B_1 , and C_1/C_1 —that is, homozygous with respect to the first alleles of genes A , B , and C . The joint probability of this occurrence for an individual belonging to population I is $(0.8)^2 \times (0.8)^2 \times (0.8)^2 = 0.26$. The joint probability of this genotype for an individual belonging to population II is $(0.2)^2 \times (0.2)^2 \times (0.2)^2 = 0.00006$. Clearly, Joe's genotype is far more common in population I than in population II, and we would conclude that Joe is more likely to be “I-ese” than “II-ian.”

Table 13.3. Example of allele frequencies for two populations measured with respect to three genes having two alleles. Both alleles of all genes are present in both populations. While the minor allele of each gene is somewhat common in each population, particular combinations of alleles may be quite rare (e.g., A_2/A_2 , B_2/B_2 , C_2/C_2 in population I).

Gene	Allele Frequencies			
	Population I		Population II	
	1	2	1	2
A	0.8	0.2	0.2	0.8
B	0.8	0.2	0.2	0.8
C	0.8	0.2	0.2	0.8

13.3.2 Population Structure

The world's human population displays **population structure**: there are clear subpopulations indicated by the data (eg, locus D12S2070, Fig. 13.1). The populations are hierarchical in the sense that each local population can be grouped into successively larger inclusive groupings (local population \rightarrow regional population \rightarrow world population). Because the local populations are genetically distinct, the total population for the world as a whole is said to be **stratified**. For such populations, there are particular relationships between heterozygosities calculated as averages of subpopulation data and heterozygosities calculated from pooled data. Note in Table 13.2 that the heterozygosity computed from population values, $H_{R,\text{avg}}$, is always less than H_R , the heterozygosity computed by pooling all individuals in a region (the region as a

whole). This type of behavior is a general occurrence when a population (e.g., in a region) is broken down into subpopulations (the individual populations in the example above). This can be shown as follows.

Suppose we wish to compare the heterozygosities for a total population, which is broken down into B subpopulations, to the average heterozygosity of the subpopulations. We perform measurements on one gene having K alleles. We let p_i be the average fraction of allele i in the population as a whole and p_{ib} be the fraction of allele i in subpopulation b . The total population homozygosity, F_T , is

$$F_T = \sum_{i=1}^K p_i^2,$$

and the homozygosity of subpopulation b is

$$F_b = \sum_{i=1}^K p_{ib}^2.$$

The average homozygosity for the subpopulations, F_S , is

$$F_S = \frac{1}{B} \sum_{b=1}^B F_b$$

(analogous to (13.3)). We are interested in the difference between F_S calculated by averaging the F_b for the subpopulations, and F_T calculated from pooled data:

$$F_S - F_T = \frac{1}{B} \sum_{b=1}^B \sum_{i=1}^K p_{ib}^2 - \sum_{i=1}^K p_i^2. \quad (13.6)$$

In Exercise 3 we show that

$$F_S - F_T = \sum_{i=1}^K \left[\frac{1}{B} \sum_{b=1}^B (p_{ib} - p_i)^2 \right]. \quad (13.7)$$

The terms in the brackets are seen to be the variances of the frequencies (across subpopulations) of each allele, measured with respect to the population average. In terms of variances σ_i^2 ,

$$F_S - F_T = \sum_{i=1}^K \sigma_i^2. \quad (13.8)$$

In terms of heterozygosities, $H_T = 1 - F_T$ and $H_S = 1 - F_S$, so we finally obtain

$$H_T - H_S = \sum_{i=1}^K \sigma_i^2. \quad (13.9)$$

Since the variances σ_i^2 are always positive, the heterozygosity for a pooled population is always larger than the average of the heterozygosities of the subpopulations of which it is constituted.

The quantities we have been discussing are closely related to the F -statistics defined by Wright (1951). These statistics were called fixation indices, and they describe the heterozygosities of structured populations at different levels,

$$F_{ST} = \frac{H_T - H_S}{H_T},$$

$$F_{SR} = \frac{H_R - H_S}{H_R},$$

$$F_{RT} = \frac{H_T - H_R}{H_T},$$

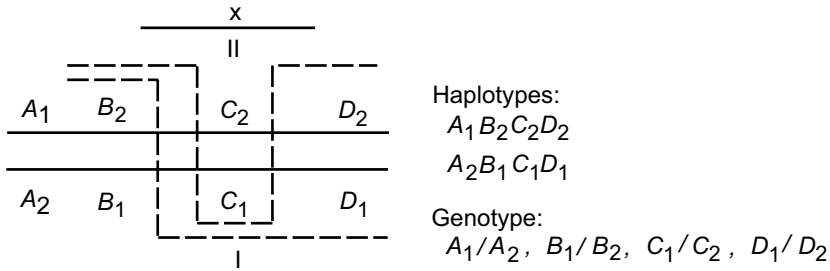
where subscripts S , R , and T stand for subpopulation, region, and total in the same sense that we have been using them above. These subscripted quantities should not be confused with homozygosities, F . Note that the terms in (13.5) are related to the fixation indices.

13.4 Effects of Recombination

The discussion up to now has not included recombination, which permits alleles on one chromosome to substitute for alleles on other copies of that chromosome (Section 1.3.2). When describing a population in terms of individual loci, it suffices to report allele frequencies corresponding to each locus. Sometimes we are interested in several different loci that reside on the same chromosome. The specification of the alleles for loci on the same chromosome is called the **haplotype**. The distinction between genotype and haplotype is shown in Fig. 13.2. Without recombination, new haplotypes would be generated only by new mutations. However, new haplotypes are produced by recombination because chromosome pairs usually recombine during gamete formation.

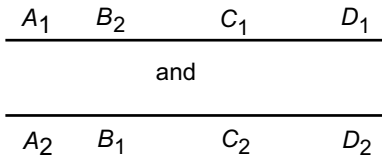
Given a group of individuals affected by a particular genetic disease phenotype, how do we identify the gene or set of genes that confer that phenotype? One way is to identify linkage of the locus associated with the disease with mapped genetic markers. Such **linkage analysis** may be performed by using families containing affected individuals. One limitation of such an approach is that the numbers of individuals in families available for genetic study are usually rather small, which means that the number of meioses represented in the pedigree (family tree) is correspondingly small. As we shall see in the next section, low numbers of meioses mean that there will be a low number of recombination events, which means that any linkage detected will probably be to genetic markers that are comparatively far away from the locus causing

A.



B. Gamete chromosomes

Single cross-over (I):



Double cross-over (II):

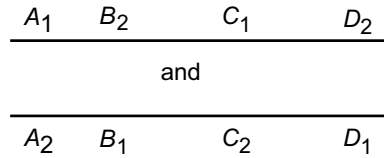


Fig. 13.2. Distinctions between haplotypes and genotypes, and the effects of recombination. Panel A: A portion of two chromosomes is shown containing four genes, each of which has two alleles. Two different recombination events (broken lines) are diagrammed. Recombination I involves a single crossover, while recombination II involves a double crossover. The products from these events are illustrated in Panel B.

the disease. Another limitation is that the interpretation of linkage analysis data is tied to a model of inheritance. An alternative to linkage analysis is **association analysis**, which seeks statistical relationships between alleles within a larger population and the disease phenotype. An example of association analysis is the case-control study. The population can now be much larger than one or a few families, so many meioses usually will have occurred since the appearance of the disease-causing allele in the population. If a particular allele of gene X is usually co-inherited with the disease state, then the disease locus is probably close to gene X. To apply this logic, we need to know the sizes of genomic regions for which allelic combinations have not been disrupted by recombination. Such regions (called haplotype blocks) are discussed in Section 13.6.

13.4.1 Relationship Between Recombination and Distance

We first develop an expression that relates the probability of recombination to distance. Two crossover events are diagrammed in Fig. 13.2. Crossover I involves a single genetic exchange in interval x and joins the left segment of the top chromosome to the right segment of the bottom chromosome and vice versa. An odd number of crossover events anywhere in region x will produce products that are recombinant for markers to the left and right of region x , in this case alleles of genes A and D . Crossover II involves two genetic exchanges in region x . Even numbers of exchanges may generate products recombinant for genes that lie to the left (or right) of x and genes lying *within* x , but they do not generate products recombinant for genes that *flank* region x . We are interested in how frequently haplotypes are shuffled by genetic exchange. The **recombination fraction** r is the probability that alleles at two loci on a chromatid come from *different* parental chromosomes. The recombination fraction, which is used to measure genetic map distance, is related to m , the expected number of crossovers between the two loci.

The simplest model relating r to m was first proposed by Haldane. He assumed that the crossovers occurring between two loci on a given chromatid follow a Poisson process with mean m . In particular, the number C of crossovers has a Poisson distribution, with

$$\mathbb{P}(C = k) = \frac{m^k e^{-m}}{k!}, \quad k = 0, 1, \dots$$

As we can see from Fig. 13.2, odd numbers of crossovers lead to recombination between loci on both sides of x (A and D), and even numbers of crossovers do not. Thus the relevant probability is

$$r = \mathbb{P}(C \text{ is odd}) = \sum_{k \text{ odd}} \frac{m^k e^{-m}}{k!}. \quad (13.10)$$

The sum on the right-hand side of (13.10) can be expressed as

$$\sum_{k \text{ odd}} \frac{m^k}{k!} = \frac{e^m - e^{-m}}{2}.$$

(This can be verified by performing series expansions of the exponential terms on the righthand side.) Therefore,

$$r = \mathbb{P}(C \text{ is odd}) = \frac{1}{2}(1 - e^{-2m}). \quad (13.11)$$

This is the Haldane mapping function, which relates recombination frequency (probability) with map distance, m . When m is small, we can neglect terms of order m^2 and higher in a series expansion of the exponential term in (13.11) to see that for short distances $r \approx m$. Note that as m gets very large in (13.11),

the probability of recombination approaches 0.5. This means that relative to a chosen locus on one chromosome, there is a 50:50 chance that an allele at a distant locus has become associated with the partner chromosome. In such a case, the loci appear to be **unlinked**. (Note that because of random assortment of chromosomes during meiosis, the probability that genes appearing on *different* chromosomes will appear in the same gamete is also 0.5.)

Loci that are close together are therefore less likely to be separated by recombination than are loci that are far apart, and for sufficiently short distances, there is an approximately linear relationship between recombination frequency and distance. This is the rationale for measuring genetic distances in terms of recombination frequencies, and the map units used for eukaryotes are **centimorgans** (named after Thomas Hunt Morgan). One centimorgan distance between two loci corresponds to having (on average) one recombination event separating these loci every 100 meioses. Because meiosis is different in male and female mammals (oocytes are arrested in meiosis I in females), the relationship between map distances in human males and females is different. In males, it is estimated that there are 0.92 cM per Mb, but for human females, there are 1.7 cM/Mb (Yu et al., 2001). The sex-average relationship between recombination frequency and distance is thus very approximately 1 cM/Mb. These numbers are only averages: recombination rates vary from chromosome to chromosome and at different positions in any particular chromosome. Local recombination rates measured at genetic map resolution range from 0.1 cM/Mb to 3 cM/Mb for Chromosome 3 (Kong et al., 2002). In some regions rates are even larger, as described in Section 13.6. The total genetic map length of an organism can be obtained by multiplying the number of cM/Mb by the physical length, or more accurately by concatenating map distances separating all loci. Using the latter approach, the genetic length of the human female autosomal genome (i.e., excluding the X chromosome) is estimated to be 4821 cM, and the corresponding length for the human male autosomal genome is about 2590 cM (Kong et al., 2002).

13.4.2 Genetic Markers

Appropriate genetic markers that can easily be assayed are required for genetic mapping or measurement of recombination. If recombination is to be detected between two loci, there must be more than one allele for each locus. For example, the recombination events I and II shown in Fig. 13.2 (bottom) produced new haplotypes that are different from the parental haplotypes. Given suitable assays, these haplotypes could be detected. However, suppose that the genotype for the rightmost marker was D_1/D_1 : if we were scoring recombination using loci A and D , we would not be able to detect recombination because the state of the allele at position D (D_1/D_1) would remain unchanged even if recombination occurred. What is required are **polymorphic markers**—genes or loci that have two or more alleles with sufficiently high frequency in the population. We say that a locus is polymorphic if it has

at least two alleles with frequency greater than 1/100. (This cutoff is arbitrary, serving only to remove alleles with extremely small frequencies.)

There are three major considerations affecting the utility of any type of genetic marker: (a) the *density of the marker* along the genome, (b) the *type of assay* of marker alleles, and (c) *amenability to high-throughput screening*. Genes were the earliest markers employed in genetics. With estimates of 25,000 genes in the human genome, the marker density is roughly 1/10⁵ bp. Traditionally, genes were scored by different methods, depending upon the phenotype that they confer. For example, different blood group alleles might be distinguished by immunological assays. Genes contributing to complex traits might have no corresponding simple phenotypic assay. If different assay methods are needed for scoring different loci, prospects for high-throughput screens based on gene products are poor.

Recombinant DNA technologies provided assays for new types of genetic markers that could be scored using the same experimental platform. One approach is gel electrophoresis of DNA digested by a particular restriction endonuclease followed by Southern blotting (i.e., transfer of resolved DNA fragments to membranes for hybridization to a labeled probe). One type of marker that can be detected in this way is the **restriction fragment length polymorphism (RFLP)**, which arises when a particular restriction endonuclease cleavage site is inactivated (i.e., mutated) in a subpopulation. Another type of polymorphism is the presence or absence of a **transposable element**, often Alu elements for human DNA. There are about 10⁶ Alu sequences in the human genome, but different human populations may display characteristic Alu insertions or deletions, which alter sizes of restriction fragments from particular chromosomal regions. Thus, Alu elements are another source of RFLPs. Microsatellite and minisatellite polymorphisms, **variable number of tandem repeat (VNTR)** polymorphisms, can sometimes be detected by gel electrophoresis and Southern blotting. These consist of tandem repetitions of DNA segments occurring at numerous locations throughout the human genome. The number of repeated bases may differ for different individuals or populations. One type (minisatellite DNA) tends to have extensive repetition of elements ranging in size from 14 to 500 bp. These are often found in telomeric regions of chromosomes. Another type (microsatellite DNA) comprises shorter blocks of sequence with repeat sizes of 1 to 13 bp. Di-, tri-, and tetra-nucleotide repeat units are common. Microsatellite DNA is distributed throughout the genome. Examples of di- and tri-nucleotide repeats are (CA)_n and (AAT)_n, with the polymorphism arising from differing values of *n* among members of the population.

The advent of the polymerase chain reaction (PCR) revolutionized genome analysis because of its sensitivity (as little as one molecule can be detected) and specificity (particular regions within a vast “sea” of genomic DNA can be selectively amplified). PCR is an automatable method that can be used to assay for VNTR and restriction site polymorphisms, obviating the need for Southern blotting (which is hard to automate and employs radioactive

probes). A disadvantage of PCR is the required design and synthesis of specific primer pairs for each region to be amplified.

A **single-nucleotide polymorphism** (SNP) is variation in the identity of the base appearing at a particular position in the genome. There are an estimated $1.5 \times 10^6 - 10^7$ polymorphic SNP sites in the human genome (Cargill et al., 1999; International SNP Map Working Group, 2001; Venter et al., 2001), corresponding to resolution at a level of hundreds of base pairs. Detection can employ either DNA sequencing or oligonucleotide array technologies, which means that SNP analysis can be readily automated and multiplexed. If detection is by hybridization, the underlying principle is that mismatched bases reduce the melting temperature of DNA-DNA hybrids. In the simplest example, four different oligonucleotide features, each having one of the four different bases at the polymorphic site, can be arrayed on a solid substrate. The state of the base at the polymorphic site is indicated by which of the four features hybridizes with the DNA being tested. An alternative detection method is primer extension. The underlying principle in this case is the ability of DNA polymerase to add the correct base to a growing chain during synthesis along a DNA template. The oligonucleotide probe sequences arrayed on a solid substrate act as primers, so designed that the next base to be added corresponds to the polymorphic site. The polymorphism can be detected by supplying dXTP molecules bearing distinctive fluorescent tags and detecting the wavelength of fluorescence emission after DNA synthesis. When the primers are not immobilized, other detection methods such as standard DNA sequencing gels or MALDI-TOF mass spectrometry may be employed.

13.5 Linkage Disequilibrium (LD)

Linkage disequilibrium (hereafter abbreviated **LD**) refers to the nonrandom association of alleles in haplotypes. It is measured by comparing the proportion of an observed haplotype with the proportion that would be predicted based upon the population frequencies of the alleles at each locus. In this section, we discuss a number of properties and consequences of LD.

13.5.1 Quantitative Description of LD

Linkage disequilibrium may be quantified as the difference between the observed and predicted frequencies for allele combinations at two or more loci. The predicted frequencies are computed using the population frequencies of the alleles. For a system in which each locus has two alleles, let $p_{A_1B_1}$ and $p_{A_2B_2}$ be the probabilities in the population of haplotypes A_1B_1 and A_2B_2 , respectively. The allele frequencies of the genes contributing to these haplotypes are p_{A_1} , p_{B_1} , p_{A_2} , and p_{B_2} . Note that the allele frequencies are population quantities, which are unaffected by recombination. We introduce a quantity

D that measures the difference between these observed and predicted frequencies:

$$D = p_{A_1B_1} - p_{A_1}p_{B_1}.$$

Similarly (see Exercise 4),

$$D = p_{A_2B_2} - p_{A_2}p_{B_2}. \quad (13.12)$$

If the A_1B_1 and A_2B_2 haplotypes are in excess of their predicted frequencies, as indicated above, then there must be corresponding deficits in the haplotypes A_1B_2 and A_2B_1 (having probabilities $p_{A_1B_2}$ and $p_{A_2B_1}$, respectively):

$$\begin{aligned} p_{A_2B_1} - p_{A_2}p_{B_1} &= -D, \\ p_{A_1B_2} - p_{A_1}p_{B_2} &= -D. \end{aligned} \quad (13.13)$$

Some straightforward algebra shows that

$$D = p_{A_1B_1}p_{A_2B_2} - p_{A_1B_2}p_{A_2B_1} \quad (13.14)$$

(taking into account the fact that $p_{A_1}p_{B_1} + p_{A_2}p_{B_2} + p_{A_1}p_{B_2} + p_{A_2}p_{B_1} = (p_{A_1} + p_{A_2})(p_{B_1} + p_{B_2}) = 1$). The value of D can be either positive or negative, and the sign depends on the arbitrary labeling of the alleles. If $D = 0$, the loci are said to be in **linkage equilibrium**.

The magnitude of D depends upon the probabilities of the individual alleles. For example, in the special case for which $p_{A_1} = p_{A_2} = p_{B_1} = p_{B_2} = 0.5$, the maximum value of D occurs when $p_{A_1B_1} = 0.5$ (which requires that $p_{A_2B_2} = 0.5$ and that $p_{A_1B_2} = p_{A_2B_1} = 0.0$). The value of D in this case is 0.5^2 , or 0.25. Alternatively, with the probabilities of all alleles again set at 0.5 and $p_{A_1B_1} = p_{A_2B_2} = 0$, then $p_{A_1B_2} = p_{A_2B_1} = 0.5$, and from (13.14), $D = -(0.5^2) = -0.25$.

In practice, it is convenient to express the data in terms of $|D'|$, where $D' = D/D_{\max}$ and D_{\max} is the maximum value of D . We show in Exercise 5 that

$$\begin{aligned} D_{\max} &= \min\{p_{A_2}p_{B_1}, p_{A_1}p_{B_2}\}, \text{ if } D > 0, \\ D_{\max} &= \min\{p_{A_1}p_{B_1}, p_{A_2}p_{B_2}\}, \text{ if } D < 0. \end{aligned} \quad (13.15)$$

$|D'|$ has the convenient property that $0 \leq |D'| \leq 1$, with $|D'| = 1$ corresponding to complete LD.

Another convenient measure of LD, r^2 , is obtained by dividing D^2 by $p_{A_1}p_{A_2}p_{B_1}p_{B_2}$:

$$r^2 = \frac{D^2}{p_{A_1}p_{A_2}p_{B_1}p_{B_2}}. \quad (13.16)$$

The quantity r^2 is the square of the Pearson product-moment correlation coefficient, which we encountered in Section 11.4. The equivalence between (13.16) and the definitional form for r^2 can be seen by noting that for binary

alleles of gene X (X_1 and X_2), the variance is $\sigma_X^2 = p_{X_1}p_{X_2}$. This means that the denominator of (13.16) is $\sigma_A^2\sigma_B^2$.

Another way of expressing r^2 with the use of (13.12) is

$$r^2 = \left(\frac{p_{A_1B_1}}{p_{A_1}p_{B_1}} - 1 \right) \left(\frac{p_{A_2B_2}}{p_{A_2}p_{B_2}} - 1 \right). \quad (13.17)$$

The quantity r^2 measures the correlation between alleles at the two sites. For markers that have complete disequilibrium and that have the same allele frequency, it takes the maximal value 1.0 ($= [(0.5/0.25 - 1)]^2$). If there is complete equilibrium, the numerators and denominators of the ratios in (13.17) are equal and $r^2 = 0$.

13.5.2 How Rapidly Does LD Decay?

LD decays over time. It is also possible to obtain an expression for D at any generation in a population in terms of the recombination parameter (recombination fraction) r between two markers. Suppose that we want to calculate the change in frequency of a haplotype in going from one generation to the next. Consider a particular A_1B_1 haplotype. This could have arisen as a non-recombinant copy of an A_1B_1 haplotype in the previous generation (frequency $(1 - r)p_{A_1B_1}$) or as a result of recombination in the previous generation between two haplotypes that were A_1* and $*B_1$ (the $*$ denoting that the allele at the other locus is immaterial). The frequency of these latter events is just $r p_{A_1}p_{B_1}$. Combining the possibilities, we see that the fraction $p'_{A_1B_1}$ of haplotype A_1B_1 is

$$p'_{A_1B_1} = (1 - r)p_{A_1B_1} + r p_{A_1}p_{B_1}. \quad (13.18)$$

If we subtract $p_{A_1}p_{B_1}$ from both sides of (13.18), we get

$$p'_{A_1B_1} - p_{A_1}p_{B_1} = (1 - r)p_{A_1B_1} - (1 - r)p_{A_1}p_{B_1},$$

which, with the definition of D in (13.12) and the fact that the allele frequencies do not change with time for a simple recombination model, becomes

$$D' = (1 - r)D. \quad (13.19)$$

This recursion formula can be applied over t generations to relate the LD D_t at generation t to the initial linkage disequilibrium D_0 :

$$D_t = (1 - r)^t D_0. \quad (13.20)$$

Equation (13.20) indicates how linkage disequilibrium changes over time. LD will decrease to 0 after a large number of generations because $0 < r \leq 0.5$. We can estimate the number of generations necessary to reduce LD to half of its present value. We take $r = 0.01$ (corresponding to 1 cM—conveniently close linkage for association studies). We solve the equation

$$\frac{D_t}{D_0} = \frac{1}{2} = (1 - r)^t$$

for t , with $r = 0.01$. The result is $t \sim 69$. Taking the human generation time to be 20 years, we calculate that the human LD half-life is about 1400 years for markers separated by 1 cM.

13.5.3 Factors Affecting Linkage Disequilibrium

In the previous section, we modeled LD using a population model that allowed recombination to occur. Other forces acting on populations can lead to association of alleles. For example, smaller populations will experience greater sampling variation, leading to *genetic drift* (Section 13.7.1). Effects of genetic drift are balanced by recombination (which breaks up haplotypes) and mutation (which introduces new haplotypes). The magnitude of LD between two polymorphic loci in a stable population is expected to depend on population size, recombination fraction, and mutation rate. Moreover, LD can also reflect the precise mutational history of a population, *even if no recombination occurs*. Consider, for example, a haplotype $A_1B_1C_1$ in an ancestor, with distances $A - B$ and $B - C$ being identical. Now assume that a particular set of mutations occurs in the pedigree shown in Fig. 13.3. The computed values of LD for loci AB and BC are not the same, even though no recombination has occurred (see Exercise 6).

Another force operating is the *founder effect*, which is observed with a small population that has grown rapidly, largely in isolation. For example, the population of Finland (now about 5×10^6) rapidly grew from a founding population of about 1000 individuals 2000 years ago. If there is rapid population expansion over a short interval, linkage disequilibrium will be higher than for a population of comparable size that has been in existence for a long period of time.

Natural selection also can affect the association of alleles in haplotypes if the individual's genotype influences reproductive fitness. If the effect on fitness of an allele at one locus is changed by an allele at another locus, there will be a disproportionate increase in the frequency of individuals with both, at the expense of individuals having only one of the alleles.

Finally, LD can be affected by the mixing of populations having different allele frequencies. For example, suppose that we had a composite population composed of three different subpopulations having the allele frequencies and population sizes shown in Table 13.4. Suppose further that each of the three subpopulations have individually reached equilibrium with respect to alleles A and B (i.e., there is no mating *between* subpopulations). By taking weighted averages, we can obtain the proportions of alleles within the composite population: $p_A = 0.0923$, $p_B = 0.177$, and $p_{AB} = 0.0277$. From (13.12), we calculate that $D = p_{AB} - p_A p_B = 0.0114$. In other words, even though the D values for each of the subpopulations were 0.0, because of population

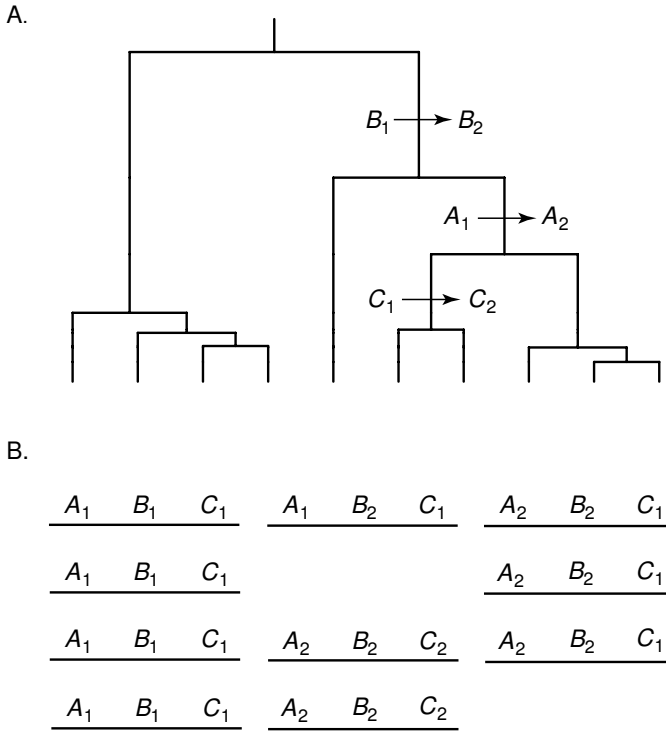


Fig. 13.3. Measures of linkage disequilibrium are dependent upon the genealogical history of mutations, independent of any recombination. A particular genealogical tree for ten haplotypes is shown in panel A. The haplotypes corresponding to the leaves of the tree immediately above each column are diagrammed in panel B.

structure (stratification), the computed D for the composite population was not zero.

Table 13.4. Example of effects of population stratification on estimated linkage disequilibrium. Probabilities of allele A at locus A and allele B at locus B are reported. Each individual subpopulation is at linkage equilibrium.

		Population				
Population	size N	p_A	p_B	p_{AB}	D	
I	1,000	0.3	0.5	0.15	0	
II	2,000	0.2	0.4	0.08	0	
III	10,000	0.05	0.1	0.005	0	
I+II+III	13,000	0.0923	0.177	0.0277	0.114	

We conclude then that the state of allelic associations in a population is the result of interplay among many aspects of the evolutionary history of the population. Unless the history is “known,” it is hard to model LD between pairs of loci. This means that in general, data for the present state of LD in populations are insufficient for making inferences about parameters such as recombination fraction, population size, or mutation rate.

13.6 Linkage Disequilibrium in the Human Genome

There are two major reasons for interest in the pattern of linkage disequilibrium in the human population. First, there are medical reasons, as indicated in Section 13.4. Recall that identification of genes associated with genetic diseases depends upon relating the disease phenotype with readily scorable genetic markers, either by using pedigree analysis or through statistical association. If there are few genetic markers on the map, disease states associated with any one of them may not be physically close, meaning that the hunt for the disease gene must cover an extensive area. Using markers such as microsatellite repeats has produced genetic maps with a resolution of about 0.5 cM (Kong et al., 2002). Association of such markers with a disease gene would (on average) allow us to restrict the search to a 500,000 bp region of the genome. The numerous single-nucleotide polymorphisms in the human genome provide an opportunity for an even finer scale of mapping (hundreds to thousands of bp map resolution). However, if there is substantial LD, a much smaller number of “tag SNPs” may serve to mark regions in linkage disequilibrium, which in turn may contain the disease-causing alleles. The second major reason is that patterns of LD reflect the evolutionary history of human populations over the last 250,000 years (e.g., effects of selection, migration, population expansion, and admixture). Different populations might have different patterns of LD, which would reflect the population history.

The ability to solve these problems has been augmented by DNA sequencing technologies and the consequent ability to detect single-nucleotide polymorphisms (SNPs). Remember that SNPs are the character states at individual nucleotide positions in a genomic sequence. In most cases examined so far, SNPs are **biallelic**: there are just two alleles, a major and a minor allele. This means that we can employ the equations developed in Section 13.5 to describe LD in the human genome. When we inquire about LD patterns in a genome, we are really asking about the frequency distribution of chromosomal lengths having $|D'|$ or r^2 values exceeding some arbitrary threshold value. These patterns could equivalently be described in terms of recombination frequencies within intervals separating the genes.

It is helpful to think about LD patterns in terms of some simple models. For example, suppose that the recombination parameter r is approximately constant throughout the length of a chromosome. In that case, we might model the locations of recombinational crossovers as a Poisson process, which would

produce an exponential distribution of lengths for segments in linkage disequilibrium if all recombination events were to occur in a single generation. Such a model applied over several generations will produce regions with high LD: loci in close proximity are less likely to be separated by recombination than distant loci, and for close loci, there simply may not have been enough meioses during the history of the population to produce short-range equilibrium. This is illustrated in Fig. 13.4. Alternatively, we might try to imagine some simple patterns to which the observed data might conform. For example, we might imagine that the genome is divided into **haplotype blocks**, which consist of consecutive chromosomal loci showing high levels of local LD, separated by short regions within which many recombination events have occurred. This is equivalent to a **punctate recombination** model (where the recombination fraction r is not constant). Whether or not these or more complicated models are appropriate, we want statistics to describe the observed patterns. For the haplotype block model, these might be (a) mean block size, (b) numbers of blocks per Mb in a region, and (c) the fraction of the region contained in (or covered by) haplotype blocks.

Descriptions of LD in terms of haplotype blocks are influenced by a number of different factors:

- The history of the sampled populations (Section 13.5.2): Populations sampled after a bottleneck will display greater amounts of LD, and populations that have experienced exponential growth will have lesser amounts of LD, than populations that have continued at stable population sizes for a long period of time. As noted above, pooled samples may produce LD by admixture.
- Sample size: The estimated haplotype frequencies are only a sample from a larger population, and estimates of the population frequencies from the sample frequencies are less reliable when the number of genomes sampled is small.
- Methods for determining haplotypes: Haplotypes may be determined directly, but often they have been inferred by statistical methods based upon allele frequencies in the sample (i.e., they are estimated from diploid genotypes). Such estimates may be less reliable than direct measures, because of required assumptions in a population model.
- Spacing between SNPs: If the marker density is high (SNPs have close spacing), short segments in LD can be detected, but if marker densities are low, only larger haplotype blocks will be observed.
- Cutoffs imposed to exclude rare alleles: To simplify interpretation of patterns, some investigators employ only more common alleles to define haplotypes. But rare alleles are found in the “younger” haplotypes, and common alleles are biased toward older haplotypes. Older haplotypes have had more time for recombination to occur, so that the length distribution of haplotype blocks may be skewed toward lower values than in younger haplotypes.

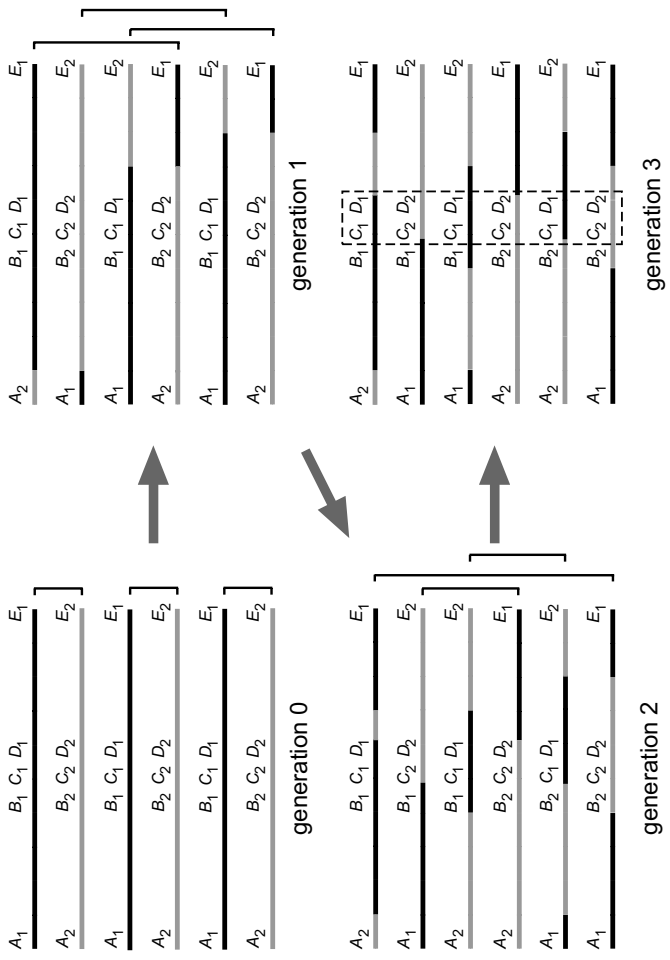


Fig. 13.4. Relationship between recombination and haplotype blocks for a population represented by six chromosomes. Initial chromosomes (generation 0) having alleles A_1, B_1, \dots, E_1 are indicated by grey lines, and initial chromosomes having alleles A_2, B_2, \dots, E_2 are shown by black lines. Chromosomes for subsequent generations are produced by recombination within all subintervals of chromosomes (brackets). In this simplified model, the probabilities of recombination within all subintervals of the same length are identical. After several generations, more crossovers between distant markers will have occurred than between close markers. Thus, haplotypes of close markers (C_1D_1 and C_2D_2 in this example) are less likely to be disrupted, leading to higher linkage disequilibrium. Region of the CD haplotype block is indicated by the box.

The observations above imply that any statistic describing LD in a particular case is meaningful only within the context of the methods and sampling protocols employed for making the estimate.

Computational Example 13.2 illustrates how SNPs can be used to identify genomic regions that display LD. In this illustration, there are five SNP loci, and the state of each is represented by one of only two alleles. Invariant loci are indicated by dashes in this illustration, but in the literature, SNP characters are usually listed in positional order with no representation of the positions between the SNPs. The measure of LD employed is $|D'|$, calculated for all possible pairings of alleles [$n(n - 1)/2$ pairings of n alleles]. The result is displayed at the end of the box in a triangular grid, with each element shaded to represent the magnitude of $|D'|$ for the corresponding pair of alleles. If the order of SNPs representing rows and columns is as shown, then the elements representing LD between SNP $j + 1$ and SNP j appear in order along the diagonal set of elements (reading from bottom left to top right). Haplotype blocks would appear as clusters of shaded elements near the diagonal. (Only one block, comprised of SNPs 3 and 4, is illustrated in this simplified example.) Figure. 13.5 presents the results for real data.

Computational Example 13.2: Calculation of pairwise linkage disequilibrium statistic $|D'|$

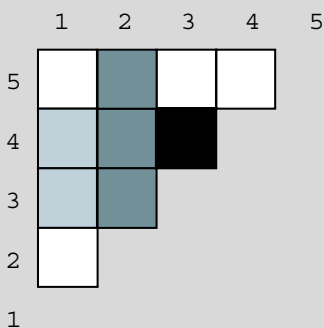
Below are the SNP data for our example.

		SNP number				
		1	2	3	4	5
--A-----	-----T---		G	T		T
--T-----	-----T---		G	T		T
--A-----	-----A--		G	T		A
--T-----	-----T--		G	T		A
--A-----	-----T--		G	T		A
--T-----	-----T--		C	A		T
--A-----	-----A--		C	A		A
--T-----	-----A--		C	A		A
--A-----	-----T--		C	A		T
--T-----	-----A--		C	A		A

The calculations are shown below.

		D	D_{\max}	$ D' $
[1, 2]	$p_{AT} = 0.3, p_A = 0.5, p_T = 0.6$	0.00	0.20	0.00
[1, 3]	$p_{AG} = 0.3, p_A = 0.5, p_G = 0.5$	-0.05	0.25	0.20
[1, 4]	$p_{AT} = 0.3, p_A = 0.5, p_T = 0.5$	-0.05	0.25	0.20

		D	D_{\max}	$ D' $
[1, 5]	$p_{AT} = 0.2, p_A = 0.5, p_T = 0.4$	0.00	0.20	0.00
[2, 3]	$p_{TG} = 0.4, p_T = 0.6, p_G = 0.5$	-0.10	0.20	0.50
[2, 4]	$p_{TT} = 0.4, p_A = 0.6, p_T = 0.5$	0.10	0.20	0.50
[2, 5]	$p_{TT} = 0.4, p_T = 0.6, p_T = 0.4$	0.16	0.24	0.67
[3, 4]	$p_{GT} = 0.5, p_G = 0.5, p_T = 0.5$	-0.25	0.25	1.00
[3, 5]	$p_{GT} = 0.2, p_G = 0.5, p_T = 0.4$	0.00	0.20	0.00
[4, 5]	$p_{TT} = 0.2, p_T = 0.5, p_T = 0.4$	0.00	0.20	0.00



White: $|D'| \leq 0.2$
 Light Grey: $0.2 < |D'| \leq 0.5$
 Dark Grey: $0.5 < |D'| \leq 0.8$
 Black: $0.8 < |D'| \leq 1.0$

Currently, data on LD in human populations has focused either on genome samples or short chromosomes (Hsa19, Hsa20, Hsa21, and Hsa22). “Haplotype block” may be defined differently in different studies. For example, haplotype blocks have been defined as a series of three or more markers in a contig for which all estimated values of $|D'|$ exceed 0.9 (Phillips et al., 2003) or as a region over which fewer than 5% of the markers show evidence of recombination (Gabriel et al., 2002). The study of Hsa19 (Phillips et al., 2003) indicated that with a median 5.5 kb spacing between SNPs, about 32% of the chromosome was contained in haplotype blocks. The observed distribution of block lengths resembled an exponential distribution, and simulations indicated that similar distributions could be produced without invoking selection, special population histories, or recombination hotspots. The median block length was approximately 20 kb. In the case of Hsa21, individual chromosomes were separated by using somatic cell hybrids, so that haplotypes were measured directly (Patil et al., 2001). The mean SNP spacing was 1300 bp, and an optimization method was employed to define blocks such that the entire genome was covered. About 80% of the chromosome was contained in blocks defined by three or more SNPs, and the mean size of these blocks was about 16 kb. The mean size of all blocks (including smaller ones) was 7.8 kb. The dependence of inferred haplotype block sizes on marker spacing is shown in Table 13.5. This

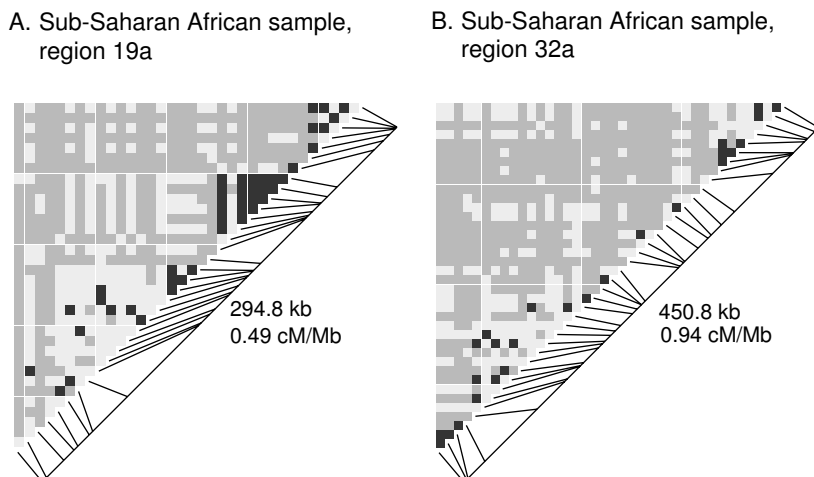


Fig. 13.5. Plots of pairwise $|D'|$ for two different human genomic regions drawn from a sample of sub-Saharan African populations. Plotting is analogous to the diagram in Computational Example 13.2. Dark grey squares indicate strong LD, light grey represents intermediate LD, and medium grey indicates no LD. Markers and marker spacings are indicated by lines below and to the right of the triangular grids. Reprinted, with permission, from Wall JD and Pritchard JK (2003) *Nature Reviews Genetics* 4:587–597. Copyright 2003 JD Wall.

emphasizes the assertion that marker spacings that are too large will fail to detect smaller blocks of disequilibrium, in effect sampling only the tail of the LD block length distribution function.

The large number of SNPs allows estimation of recombination rate variation across the human genome. Although Innan et al. (2003) found that the observed LD for chromosomes Hsa19 and Hsa21 could be modeled by using uniform but lower-than-average recombination rates, other experiments indicated that recombination “hotspots” are present in the Class II region of the major histocompatibility complex (MHC) in humans (Jeffreys et al., 2001; Kaupi et al., 2003). Substantial LD and apparent haplotype blocks were noted based upon genetic data alone, and recombination events were detected experimentally by assaying SNPs in DNA extracted from single sperm cells. Five recombination hotspots were identified, each extending for 1–2 kb and lying between the observed haplotype blocks. Whereas the average genomic recombination rate is about 1.1 cM/Mb, the rates at these hotspots range from about 3 cM/Mb up to 130 cM/Mb—100- to 1000-fold higher than rates within the haplotype blocks themselves. A more comprehensive statistical study employing SNP data for Hsa20 indicated that punctate recombination (i.e., presence of hotspots and coldspots) is a general feature of the human genome (McVean et al., 2004). This study found that half of all recombination events occur within just 10% of the genome sequence and that the average

spacing between hotspots is no more than 200 kb. Recombination rates were as low as 0.01 cM/Mb for coldspots and approached 100 cM/Mb for hotspots.

Table 13.5. Effects of marker spacing on predicted haplotype block length (Phillips et al., 2003).

Distance between markers (kb)	Predicted mean block length (kb)
0.1	0.5
1.0	4.0
5.0	16.8
10.0	32.4
50.0	148.6

As we mentioned above, it is estimated that there are some $1.5 \times 10^6 - 10^7$ common SNP loci in the human genome, a number far too large to be economically feasible for routine screening of individuals. LD structure offers the opportunity to reduce the labor while still reaping the benefit of having more closely spaced markers than are available on current genetic maps. With regions of substantial LD extending for about 20 kb (see above) and the assumption that most of the genome lies within such blocks, there would be about 150,000 such regions in the human genome. If we were to use a corresponding number of tag SNPs, then the resolution of the human genetic map would be improved to about 0.02 cM ($3700 \text{ cM (average)} / 150,000$), while retaining a manageable number of markers to screen for association with genes implicated in genetic diseases.

13.7 Modeling Gene Frequencies in Populations

13.7.1 The Wright-Fisher Model

To understand the effects of different forces on the distribution of gene frequencies in populations, it is helpful to use stochastic or deterministic models. We saw an example of the latter type when we discussed the rate of decay of LD in Section 13.5.2. In this section, we introduce some Markov chain models to describe gene frequencies in finite populations.

We begin with the simplest model, which was introduced by Fisher (1930) and Wright (1931). The assumptions we make to build this model are:

- The population size N is constant from generation to generation.
- Organisms are diploid (so there are $2N$ copies of each gene).
- All members of each generation reproduce simultaneously: generations do not overlap.

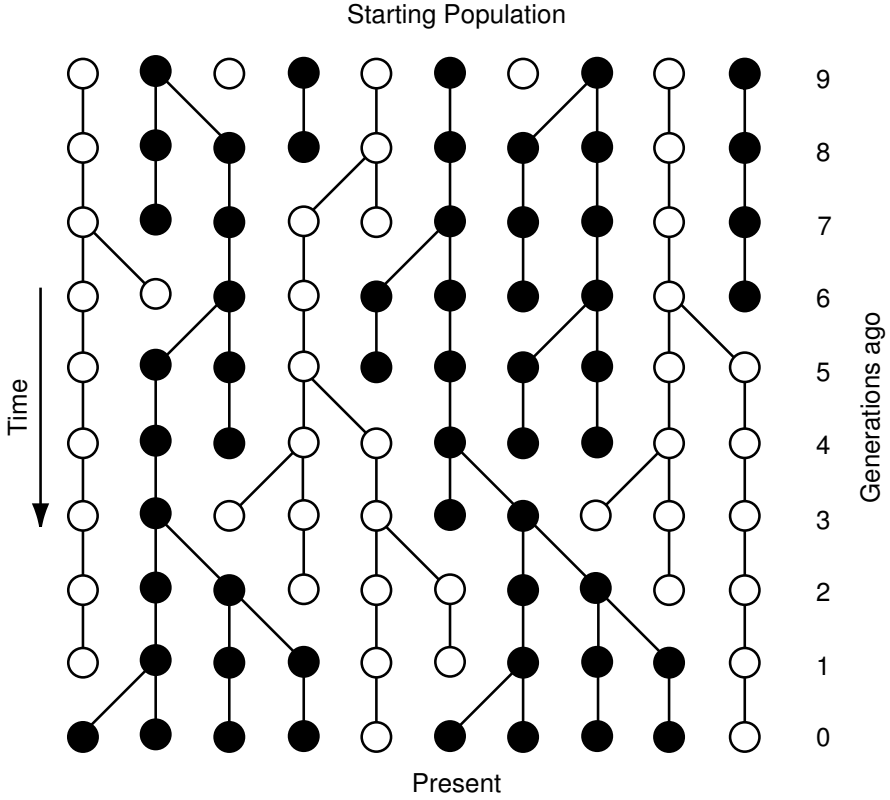


Fig. 13.6. Schematic illustration of Wright-Fisher model for a population of size $N = 5$, with two alleles ($A =$ white circles, $B =$ black circles) at equal starting frequencies. Each horizontal line of $2N = 10$ circles corresponds to a different generation. The total population size does not change from generation to generation. Copies of each gene to be reproduced for the next generation are chosen randomly, so that some may be copied more than once and others not at all. This leads to **genetic drift**, illustrated here by the excess of B alleles in the present population (0 generations ago). Note that three generations ago there was an excess of W alleles.

- Mating among individuals is random.
- Allele frequencies are not perturbed by mutation, migration, or selection.

The term “random mating” deserves further explanation. To form an offspring, choose an individual at random from the population and then choose one if its gametes at random. Return the chosen individual to the population, and repeat the experiment. This results in two gametes that form an individual in the next generation. This procedure is repeated N times to form the next generation. Our model, diagrammed in Fig. 13.6, describes a population in terms of a **gene pool** from which genes are randomly drawn to constitute

the next generation. To describe a population containing N individuals, it is not necessary to keep track of each individual. Instead, we follow the fate of the $2N$ copies of each gene. Suppose that each gene is either allele A or allele B . If at generation n there are i copies of allele A , then the number of A alleles at generation $n + 1$ has a binomial distribution with $2N$ trials and success probability $p = i/2N$.

Because of the random components in this model, allele frequencies change over time, even though the model does not allow for mutation or selection. This process is called genetic drift. The random number of offspring per individual implies that an initial population allowed to evolve over t generations will produce different outcomes for different “trials.” (In the real world, we observe populations that have experienced only one trial during the course of evolution.) The simulation below illustrates genetic drift in a population for a gene having two different alleles. Notice that the allele frequencies evolve differently for different trials. At some point, one allele or the other “wins” (**fixation** of that allele) and the other allele is lost (it is “unlucky”). The surviving allele has become fixed, not as a result of selection based upon any benefit it may confer but merely as a result of the random nature of the process.

Computational Example 13.3: Simulation of genetic drift

We can see the effects of genetic drift for a two-allele system by employing the R function below. The parameters are population size, N ; the number of alleles A in the population at the beginning, M ; and the number of generations, G . (The initial generation is labeled 1 here.) The function `drift` is employed:

```
drift<-function(N,G,M){
  # N = number of genes in population
  # G = number of generations of simulation
  # M = number of A alleles in population
  pop<-matrix(nrow=N, ncol=G)
  # Holds resulting alleles, each generation
  prop<-rep(NA,G)
  # Holds proportion of allele A in each generation
  prop[1]<-M/N
  #Initialize the first generation
  pop[,1]<-c(rep(1,M),rep(0,(N-M)))
  for(j in 2:G){ #looping over generations
    for (i in 1:N){
      pop[i,j]<-sample(pop[,j-1],1,replace=TRUE)
    }
    #####
    # Alternative code to replace interior loop: #
    # pop[,j]<-sample(pop[,j-1],replace=TRUE) #
  }
}
```

```
#####
prop[j]=sum(pop[,j])/N
}
return(prop)
}
```

We perform four different runs for 100 generations with an initial population of 60 genes, with $M = 30$:

```
> tmp<-matrix(nrow=4,ncol=100)
> for(i in 1:4){tmp[i,]<-drift(N,G,M)}
```

Plots for each run are shown in Fig. 13.7.

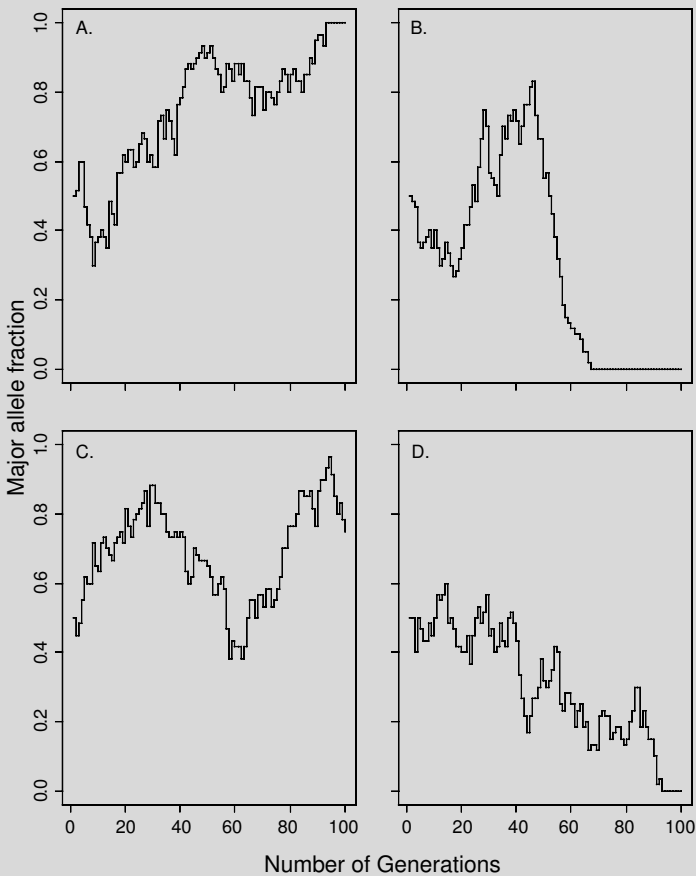


Fig. 13.7 Simulation of genetic drift after 100 generations for 60 biallelic genes, with both alleles initially at the same frequency. Four independent realizations of

this process are illustrated. One allele or the other becomes fixed in simulations A, B, and D. For simulation C, neither allele has been fixed after 100 generations. The plots were produced using:

```

> par(mfrow=c(2,2))
> plot(1:G,tmp[1,],type="s",lty=1,ylim=c(0,1.0))
# Repeat for tmp[2,],...,tmp[4,]

```

Observe that in one of the runs (panel A) the A allele has drifted to a relative frequency of 1.0, and in two of the cases (panels B and D) the A allele has become extinct. In one run (panel C), the A allele has been enriched but has not been lost or fixed in 100 generations. Notice that, in this case, the A allele almost became fixed by generation 94, after which the other allele began to experience a run of “good luck.”

13.7.2 The Wright-Fisher Model as a Markov Chain

If we denote the number of A alleles in the population in generation n by X_n , then we recognize that the sequence X_0, X_1, \dots is a Markov chain, the set of possible outcomes being $\{0, 1, 2, \dots, 2N\}$. The transition matrix of the chain (see page 52) is given by

$$p_{ij} = \binom{2N}{j} \left(\frac{i}{2N}\right)^j \left(1 - \frac{i}{2N}\right)^{2N-j}, \quad i, j = 0, 1, \dots, 2N.$$

We note that, conditional on $X_n = i$,

$$\mathbb{E}(X_{n+1} \mid X_n = i) = 2N \times \frac{i}{2N} = i,$$

so that

$$\mathbb{E}X_n = \mathbb{E}X_0 \quad \text{for all } n.$$

This result says that *on average* the A allele frequency does not change. This result is akin to the Hardy-Weinberg law, but it gives a very misleading impression of the behavior of the model. Recall that in our simulations we saw trajectories in which the A allele became fixed! Because there is no mutation in this model, the states 0 and $2N$ are absorbing—once the frequency of allele A reaches 0 or $2N$, it remains there forever after. It can be shown that the chance that allele A becomes fixed is its initial relative frequency (see Exercise 7).

We know that fixation or loss of the A allele must occur. We can find the rate at which this loss occurs by calculating the expected heterozygosity $h(n)$ in the population in generation n . This is the probability that two genes chosen at random (with replacement) are different alleles. It can be shown that

$$h(n) = \left(1 - \frac{1}{2N}\right)^n h(0). \quad (13.21)$$

Variation is therefore lost at a geometric rate, and this rate is faster in a small population than in a large one (see Exercise 8).

13.7.3 Including Mutation

The model in Section 13.7.1 does not allow for mutation. In this section, we add **neutral mutations** to the Wright-Fisher model. Neutral mutations do not affect the survival or reproductive success of an organism. Neutral mutations thus can lead to a variety of alleles that are not subject to natural selection. A subset of neutral mutations are those that occur in the third position of codons such that the specified amino acids remain unchanged.

We employ the **infinitely many alleles model**, which states that each mutation produces a novel allele. This implies that any *homozygous* individuals are also autozygous because, under this mutation model, if two alleles are the same, they must have descended from the same ancestor. This also implies that alleles cannot back-mutate. Of course, the number of alleles cannot actually be infinite, but this is a good approximation, provided the actual number of possible alleles is sufficiently large. For example, a look at a table of the genetic code shows that, given any specification of first and second positions in a codon, there are on average 2.7 third-position variants that will specify the same amino acid. For a polypeptide chain containing about 400 amino acid residues (a reasonable average for eukaryotes), there are about 2.7^{400} possible alleles involving third positions of codons and producing the same amino acid sequence. This number does not take into account amino acid composition, which would affect the average number of third position variants. This works out to approximately 3.5×10^{172} alleles that would code for the same polypeptide chain. This number is not infinite but still is very large.

We now compute some population statistics in terms of this model. As before, we track $2N$ copies of each gene at each generation. Let μ be the rate of mutation of each gene; this rate is assumed to be constant for all genes and for all generations. We obtain an expression for the probability F_t that two genes are identical by descent in generation t . This event can occur in two ways: either both genes are copies of a single gene from generation $t - 1$ (probability $1/2N$) and neither copy is mutant, or the two genes are nonmutant copies of different genes in the previous generation and those two genes are identical by descent. Thus,

$$F_t = \frac{1}{2N}(1 - \mu)^2 + \left(1 - \frac{1}{2N}\right)(1 - \mu)^2 F_{t-1}. \quad (13.22)$$

If enough generations pass, genetic drift and mutation lead to a steady-state condition in which the probability of homozygosity does not change. After

this point, $F_{t-1} = F_t = F$, a constant. Because μ in 13.22 is a very small number (around 10^{-6} amino acid substitutions per coding region per year), we can ignore terms in μ^2 . Also, since N is ordinarily a large number, we can also ignore terms in μ/N . Expanding the terms $(1 - \mu)^2$ in (13.22), applying the approximations, and rearranging gives the result

$$F = \frac{1}{1 + 4N\mu}. \quad (13.23)$$

The heterozygosity H is just $1 - F$ (by definition), so

$$H = \frac{4N\mu}{1 + 4N\mu}. \quad (13.24)$$

Equation (13.24), unlike (13.2), was derived in terms of a model, without explicit knowledge of the individual allele frequencies. As we see later, the product $4N\mu$ appears repeatedly in other contexts, so for convenience we write

$$\theta = 4N\mu. \quad (13.25)$$

We will discover in Section 13.8.2 that θ is the expected number of differences in two sequences drawn at random from a population of size N . The parameters μ and N are usually confounded—they appear together in θ as a product—and from population data from a single generation alone they cannot be estimated independently. The quantity θ is called the *mutation parameter*.

13.8 Introduction to the Coalescent

We note that (13.22) was found by asking how the relevant statistic in generation t , F_t , could have been obtained, given its value in the prior generation. In other words, rather than considering how the statistic changes as we go forward in time, we found it practical to look backward. In a sense, this is a very natural thing to do when considering the genetics of populations. We are given a population as it exists now, and we may want to make inferences about how it reached its current state. (Predicting the future is beyond the realm of observational science unless one can afford to wait a suitably long time to check predictions.) The **coalescent** (Kingman, 1982) is a very useful stochastic process that allows us to model the ancestry of genes in the population. This section describes the nature of the coalescent.

13.8.1 Coalescence for Pairs of Genes

We illustrate how the coalescent works by discussing two aspects: the time to the most recent common ancestor (T_{MRCA}) for two gene sequences from the

sampled population and the expected number of pairwise differences between a pair of sequences. We use the Wright-Fisher model. The population contains $2N$ copies of each autosome or autosomal single-copy gene at each generation. (Don't worry about mutation yet—that is dealt with at the end of the argument.) We discuss the model in terms of an autosomal gene. We count time backward from the present, so at the present $g = 0$, and successive generations back in time are at times $g = 1, 2, \dots$. (We use g to denote generation in this section, as a reminder that time is running back into the past.) Each gene in generation $g - 1$ had an ancestor in generation g . (Note the direction for the generation scale!) The coalescent model is implemented by allowing each gene in generation $g - 1$ to “choose” its ancestor from among the $2N$ gene copies that existed in generation g . Clearly, some of the gene copies in generation g may be chosen multiple times, and others may be chosen not at all. The process is repeated, going back from generation g to generation $g + 1$. Because some gene copies are not chosen in each generation going back, the number of ancestors becomes smaller and smaller until the lineages coalesce to a single ancestor some number of generations ago (Fig. 13.8). Note that the process is stochastic: this means that different “runs” of this process yield different values of the T_{MRCA} . We seek the expectation of this quantity for two genes.

Pick one gene out of the $2N$ copies in generation $g - 1$. It came from one of $2N$ ancestors in generation g . The probability that a second gene in generation $g - 1$ came from the same parent is $1/2N$. The probability that the second gene came from a different parent is therefore $1 - 1/2N$. The probability that two genes have not coalesced to a common ancestor within g generations is the product of the probabilities that they have not coalesced at generations $1, 2, \dots, g$:

$$\mathbb{P}(\text{First coalescence} > g \text{ generations}) = \left(1 - \frac{1}{2N}\right)^g. \quad (13.26)$$

We see that T_{MRCA} for a pair of genes has a geometric distribution with

$$\mathbb{P}(\text{First coalescence } g \text{ generations ago}) = \frac{1}{2N} \left(1 - \frac{1}{2N}\right)^{g-1}, \quad g = 1, 2, \dots$$

From Exercise 9b in Chapter 3, it follows that

$$\mathbb{E}T_{\text{MRCA}} = 2N.$$

When the population size N is large, we can approximate the distribution of the time to the most recent common ancestor by an exponential distribution with mean 1. To do this, we measure time in units of $2N$ generations. Then, for $t = g/2N$,

$$\mathbb{P}(\text{First coalescence} > t \text{ units ago}) = \left(1 - \frac{1}{2N}\right)^{2Nt} \approx e^{-t}.$$

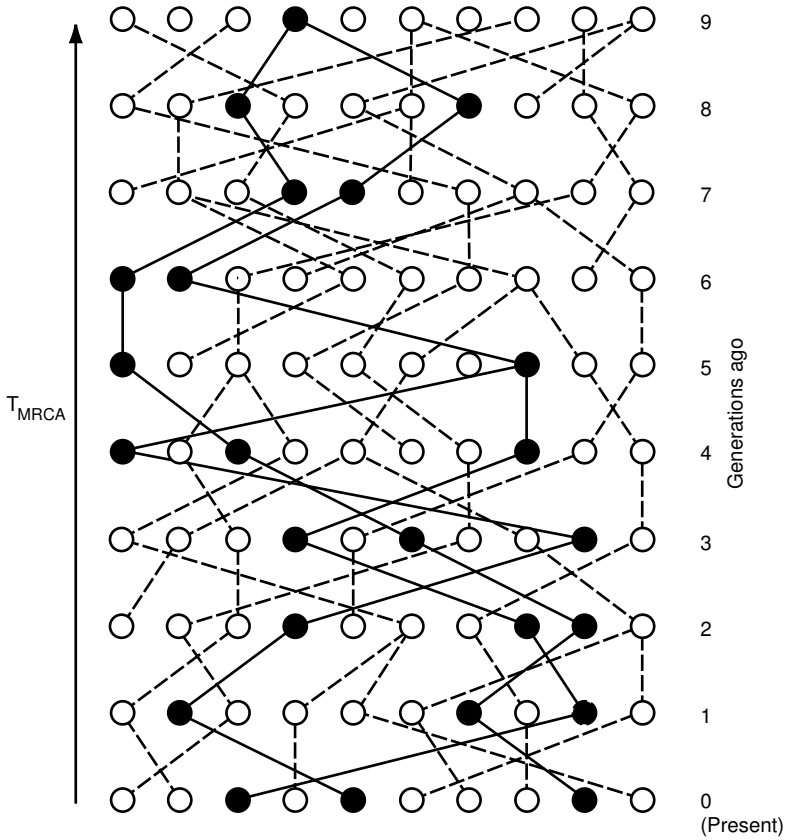


Fig. 13.8. Model for coalescence. Ten successive populations are shown. Time, measured in generations, is counted backward from the present (bottom population). Each member of population $g - 1$ “chooses” its ancestor randomly from population g . Black circles in the bottom population represent the present members of three lineages. Going back in time (increasing the number of generations g), the lineages successively coalesce until they unite at a shared common ancestor (black circle, top line). This type of analysis relates the time to most recent common ancestor, T_{MRCA} , to population size.

Thus, in a large population with time measured in units of $2N$ generations, the time to the most recent common ancestor of a pair of genes has probability density function $f_2(t)$ given by

$$f_2(t) = e^{-t}, \quad t > 0. \tag{13.27}$$

13.8.2 The Number of Differences Between Two DNA Sequences

We can use the previous results to obtain the expected number of differences between any two DNA sequences sampled in the present generation. The

number of mutations that occur along a single lineage of length g generations has a binomial distribution with parameters g and μ . With $g = 2Nt$ and $\mu = \theta/4N$ and N large, we can use the Poisson approximation to the binomial distribution to see that on our new timescale the number of mutations along a lineage of length t time units has a Poisson distribution with mean $\theta t/2$. If the two sequences have a coalescence time of T_{MRCA} , then they are separated by an amount of time equal to $2T_{\text{MRCA}}$. Given T_{MRCA} , we see that the total number of mutations separating the two sequences has a Poisson distribution with mean $(2T_{\text{MRCA}}) \times (\theta/2) = \theta T_{\text{MRCA}}$.

We use the **infinitely many sites model**, under which each mutation occurs at a site in the DNA sequence that has not had a mutation before. These sites are said to be *segregating*—the site contains two bases, the ancestral one and the mutant one. Letting Π represent the number of segregating sites in the two sequences, we see that the expected number of segregating sites is just the expected number of mutations separating the two sequences:

$$\mathbb{E} \Pi = \mathbb{E} \theta T_{\text{MRCA}} = \theta \mathbb{E} T_2 = \theta \times 1 = \theta. \quad (13.28)$$

This quantity is seen to be the same as (13.25).

One statistic commonly used to describe the variation in a set of n DNA sequences is the so-called **nucleotide diversity**. This is the average pairwise distance between the n sequences; the result in (13.28) shows that the average value of the nucleotide diversity is the compound mutation parameter θ (see Exercise 9).

13.8.3 Coalescence in larger samples

To study the ancestral relationships among a sample of size n taken from a large population we proceed as follows. The probability that the n genes have distinct ancestors in the previous generation is

$$\prod_{j=1}^{n-1} \left(\frac{2N-j}{2N} \right) = \prod_{j=1}^{n-1} \left(1 - \frac{j}{2N} \right) \approx 1 - \frac{\sum_{j=1}^{n-1} 1/j}{2N} = 1 - \frac{\binom{n}{2}}{2N}. \quad (13.29)$$

Repeating this argument, we see that the probability that no coalescence events have occurred in g generations is $(1 - \binom{n}{2}/2N)^g$. When time is measured in units of $2N$ generations, we obtain

$$\mathbb{P}(\text{First coalescence} > t \text{ units ago}) = \left(1 - \frac{\binom{n}{2}}{2N} \right)^{2Nt} \approx e^{-\binom{n}{2}t}.$$

Thus the time T_n taken for the first coalescence event in the sample has an exponential distribution with parameter $\binom{n}{2}$. It can be shown that when this event occurs, it results in the coalescence of precisely two randomly chosen members of the sample; the possibility of three or more members of the sample

coalescing simultaneously can be ignored. At that time, the sample has $n - 1$ distinct ancestors. We can repeat the previous argument to see that, with time measured in units of $2N$ generations, we wait a further amount of time T_{n-1} having an exponential distribution with parameter $\binom{n-1}{2}$, and then choose two of those $n - 1$ ancestors to coalesce. This process of randomly joining pairs of ancestors continues back to the common ancestor of the sample.

In summary, the waiting times T_n, \dots, T_3, T_2 for coalescence events are independent of each other and have exponential distributions with

$$\mathbb{E}T_j = \frac{2}{j(j-1)} \quad j = n, n-1, \dots, 2. \quad (13.30)$$

At each coalescence event a randomly chosen pair of ancestors is chosen to coalesce.

We can find the expected time to the most recent common ancestor T_{MRCA} of the sample of n genes by noting that

$$T_{\text{MRCA}} = T_n + T_{n-1} + \dots + T_2,$$

so that

$$\begin{aligned} \mathbb{E}T_{\text{MRCA}} &= \mathbb{E}(T_n + T_{n-1} + \dots + T_2) \\ &= \mathbb{E}T_n + \mathbb{E}T_{n-1} + \dots + \mathbb{E}T_2 \\ &= \frac{2}{n(n-1)} + \frac{2}{(n-1)(n-2)} + \dots + \frac{2}{2 \times 1} \\ &= 2 \left(\frac{1}{n-1} - \frac{1}{n} \right) + 2 \left(\frac{1}{n-2} - \frac{1}{n-1} \right) + \dots + 2 \left(\frac{1}{1} - \frac{1}{2} \right) \\ &= 2 \left(1 - \frac{1}{n} \right). \end{aligned}$$

Thus, the expected height of the ancestral tree linking n genes is about 2 coalescent time units. The variance of the height can be computed in a similar way (see Exercise 10). When measured on the original time scale, the expected height of the coalescent tree is $2N \times 2(1 - 1/n)$ generations, or about $4N$ generations in a large sample.

The coalescent can be thought of as a random bifurcating tree whose properties can be studied by simulation. In Fig. 13.9, some random coalescent trees for samples of size 5 are shown. Notice the large variability in the height of the trees, and observe that most of this variability is due to the deep coalescence events, those near the top of the tree. On average, over half of the height of the coalescent tree comes from the deepest coalescence.

13.8.4 Estimating the Mutation Parameter θ

Experimental estimation of recombination and mutation rates is very difficult, primarily because these rates are so small. In this section, we outline two model-based approaches for estimating mutation rates.

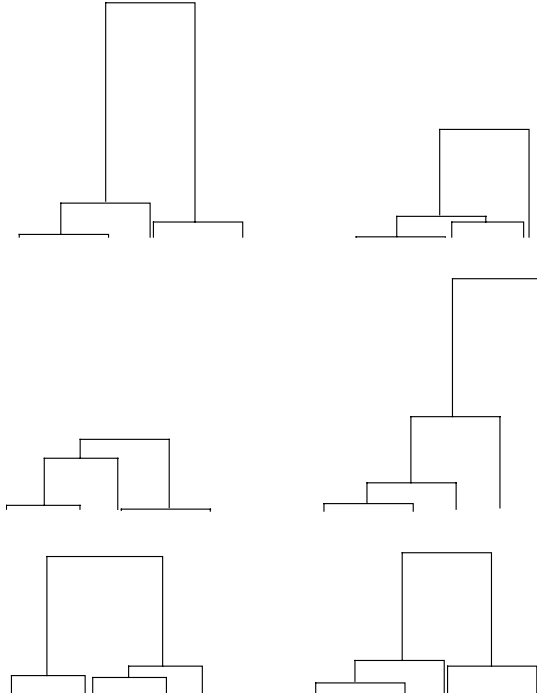


Fig. 13.9. Six realizations, drawn on the same scale, of coalescent trees for a sample of $n = 5$. (In each tree, the labels 1, 2, 3, 4, 5 should be assigned at random to the leaves.)

Under the coalescent model, mutations are placed on the tree according to independent Poisson processes of rate $\theta/2$ down each branch of the tree. Since a coalescent tree has j branches of length T_j , the length of the coalescent tree of a sample of size n is

$$L_n = \sum_{j=2}^n jT_j, \tag{13.31}$$

and the mean length, from Exercise 11, is

$$\mathbb{E}L_n = 2 \sum_{j=1}^{n-1} \frac{1}{j}. \tag{13.32}$$

If we suppose that each mutation results in a new segregating site, then we see that the number of segregating sites in the sample is precisely the total number of mutations that have arisen on the coalescent tree. The Poisson nature of the mutations means that given the length L_n of the tree, the number of mutations is Poisson with mean $\theta L_n/2$. Hence, by averaging over all possible tree lengths and using (13.32), we find that the expected number of segregating sites in our sample is

$$\mathbb{E}(\text{Number of segregating sites}) = \theta \sum_{j=1}^{n-1} \frac{1}{j}. \quad (13.33)$$

We can use the result in (13.33) to provide an estimator of the parameter θ . If we observe S segregating sites in the sample of n (aligned) sequences, then we can use the Watterson estimator

$$\theta_W = \frac{S}{\sum_{j=1}^{n-1} \frac{1}{j}}. \quad (13.34)$$

This estimator is, by design, unbiased (that is, $\mathbb{E}\theta_W = \theta$). We can compute its variance (Exercise 12a), from which we find a rather surprising result: the variance of the estimator decays at a rate proportional to $1/\log n$ (as opposed to a rate proportional to $1/n$ that would be expected for estimates of a population parameter based on an independent sample). It is precisely the dependence caused by the relatedness of genes in the sample that reduces the amount of information contained in that sample. This phenomenon is typical of estimators based on coalescent models. In Exercise 13, the properties of θ_W are studied by simulation.

An alternative approach to inference for population parameters is a Bayesian one (cf. Tavaré et al. 1997). In this approach, we find the *posterior* distribution of θ given the number of segregating sites S observed in the sample. We start by specifying a prior distribution $\pi(\theta)$ for the parameter θ , and then compute the posterior distribution $f(\theta|S)$ via Bayes' Theorem (recall (2.24)) in the form

$$f(\theta|S = k) = \frac{\mathbb{P}(S = k|\theta) \pi(\theta)}{\mathbb{P}(S = k)}. \quad (13.35)$$

Recalling (3.3), we see that the normalizing constant is

$$\mathbb{P}(S = k) = \int \mathbb{P}(S = k|\theta) \pi(\theta) d\theta.$$

All our inferences about θ are contained in the posterior density rather than point estimates such as θ_W . Finding the form of the posterior density usually requires a computational approach, one of which is outlined below.

We solve an apparently harder problem first. We simulate observations from the posterior distribution of θ and the coalescence times T_2, \dots, T_n , given that we observe $S = k$ segregating sites in the sample. Write $\mathcal{T} = (T_2, \dots, T_n)$, and note that the posterior distribution of (θ, \mathcal{T}) given $S = k$ is given by

$$f(\theta, \mathcal{T}|S = k) \propto \mathbb{P}(S = k|\theta, \mathcal{T}) \pi(\theta) f(\mathcal{T}), \quad (13.36)$$

where $f(\mathcal{T})$ is the prior probability density for \mathcal{T} obtained from the coalescent model. The likelihood $\mathbb{P}(S = k|\theta, \mathcal{T})$ can be found as follows. Writing $L = 2T_2 + \dots + nT_n$, we see that

$$\begin{aligned}\mathbb{P}(S = k|\theta, \mathcal{T}) &= \mathbb{P}(\text{Poisson with mean } \theta L/2 = k) \\ &= e^{-\theta L/2}(\theta L/2)^k/k!,\end{aligned}\tag{13.37}$$

the last coming from the form of the Poisson distribution in (3.2). A rejection algorithm for simulating observations from (13.36) is as follows:

1. Simulate an observation θ having density $\pi(\theta)$, and simulate \mathcal{T} having independent exponential distributions with means given in (13.30).
2. Calculate $L = 2T_2 + \dots + nT_n$ and $h = e^{-\theta L/2}(\theta L/2)^k/k!$.
3. Simulate U from a uniform density on $(0, 1)$. If $U \leq h$, accept the observation (θ, \mathcal{T}) if $U < h$, and ignore it otherwise. Return to step 1.

Because the probability density of accepted observations is proportional to the product of $\mathbb{P}(S = k|\theta, \mathcal{T})$ and $\pi(\theta)$, the algorithm does indeed generate observations from the posterior density in (13.36). The θ values then have the distribution (13.35) that we wanted. Implementation in R and further details are discussed in Exercise 14.

13.9 Concluding Comments

Our introduction to stochastic models of gene frequencies and estimation of population parameters such as the mutation rate θ has necessarily been very brief. In particular, we have not made explicit reference to the effects of recombination or population expansion in this setting. There is an extensive body of literature about coalescents with recombination. It provides a useful theoretical tool for interpreting patterns of LD in natural populations, and for inference about recombination rates. For a taste of this, see Nordborg and Tavaré (2002) for example.

References

- Cargill M, Altshuler D, Ireland J, Sklar P, Ardlie K, Patil N, Lane CR, Lim EP, Kalyanaraman N, Nemesh J, Ziaugra L, Friedland L, Rolfe A, Warrington J, Lipshutz R, Daley GQ, Lander ES (1999) Characterization of single-nucleotide polymorphisms in coding regions of human genes. *Nature Genetics* 22:231–238.
- Fisher RA (1930) *The Genetical Theory of Natural Selection*. Oxford: Clarendon Press.
- Gabriel SB, Schaffner SF, Nguyen H, Moore JM, Roy J, Blumenstiel B, Higgins J, DeFelice M, Lochner A, Faggart M, Liu-Cordero SN, Rotimi C, Adeyemo A, Cooper R, Ward R, Lander ES, Daly MJ, Altshuler D (2002) The structure of haplotype blocks in the human genome. *Science* 296:2225–2229.

- Innan H, Padhukasahasram B, Nordborg M (2003) The pattern of polymorphism on human chromosome 21. *Genome Research* 13:1158–1168.
- International SNP Map Working Group (2001) A map of human genome sequence variation containing 1.42 million single nucleotide polymorphisms. *Nature* 409:928–933.
- Jeffreys AJ, Kaupi L, Neumann R (2001) Intensely punctate meiotic recombination in the class II region of the major histocompatibility complex. *Nature Genetics* 29:217–222.
- Kaupi L, Sajantila A, Jeffreys AJ (2003) Recombination hotspots rather than population history dominate linkage disequilibrium in the MHC class II region. *Human Molecular Genetics* 12:33–40.
- Kingman JFC (1982) On the genealogy of large populations. *Journal of Applied Probability* 19A:27–43.
- Kong A, Gudbjartsson DF, Sainz J, Jonsdottir GM, Gudjonsson SA, Richardson B, Sigurdardottir S, Barnard J, Hallbeck B, Masson G, Shlien A, Palsson ST, Frigge ML, Thorgeirsson TE, Gulcher JR, Stefansson K (2002) A high-resolution recombination map of the human genome. *Nature Genetics* 31:241–247.
- McVean GA, Myers SR, Hunt S, Deloukas P, Bentley DR, Donnelly P (2004) The fine-scale structure of recombination rate variation in the human genome. *Science* 304:581–584.
- Nordborg M, Tavaré S (2002) Linkage disequilibrium: What history has to tell us. *Trends in Genetics* 18:83–90.
- Patil N, Berno AJ, Hinds DA, Barrett WA, Doshi JM, Hacker CR, Kautzer CR, Lee DH, Marjoribanks C, McDonough DP, Nguyen BTN, Norris MC, Sheehan JB, Shen N, Stern D, Stokowski RP, Thomas DJ, Trulson MO, Vyas KR, Frazer KA, Fodor SPA, Cox DR (2001) Blocks of limited haplotype diversity revealed by high-resolution scanning of human chromosome 21. *Science* 294:1719–1723.
- Phillips MS et al. (2003) Chromosome-wide distribution of haplotype blocks and the role of recombination hot spots. *Nature Genetics* 33:382–387.
- Rosenberg NA, Pritchard JK, Weber JL, Cann HM, Kidd KK, Zhivotovsky LA, Feldman MW (2002) Genetic structure of human populations. *Science* 298:2381–2385.
- Tavaré S, Balding DJ, Griffiths RC, Donnelly P (1997) Inferring coalescence times for molecular sequence data. *Genetics* 145:505–518.
- Venter JC et al. (2001) The sequence of the human genome. *Science* 291:1304–1351.
- Yu A et al. (2001) Comparison of human genetic and sequence-based maps. *Nature* 409:951–953.
- Wall JD, Pritchard JK (2003) Haplotype blocks and linkage disequilibrium in the human genome. *Nature Reviews Genetics* 4:587–597.
- Wright S (1931) Evolution in Mendelian populations. *Genetics* 16:97–159.
- Wright S (1951) The genetical structure of populations. *Annals of Eugenics* 15:323–354.

Exercises

Exercise 1. Calculate the heterozygosities corresponding to each of the allele probability distributions presented below.

Probabilities for allele				Heterozygosity
1	2	3	4	
1	0	0	0	?
0.8	0.2	0	0	?
0.67	0.33	0	0	?
0.5	0.5	0	0	?
0.5	0.3	0.2	0	?
0.33	0.33	0.33	0	?
0.4	0.3	0.2	0.1	?
0.25	0.25	0.25	0.25	?

Exercise 2. Prove that for a particular locus for which there are k alleles, the heterozygosity is maximized when all alleles have the same frequency.

Exercise 3. This exercise establishes the equality of (13.6) and (13.7). To see this, note that (13.6) can be rewritten as

$$F_S - F_T = \sum_{i=1}^K \left[\frac{1}{B} \sum_{b=1}^B (p_{ib}^2 - p_i^2) \right].$$

Completing the square for the term in parentheses gives

$$F_S - F_T = \sum_{i=1}^K \left[\frac{1}{B} \sum_{b=1}^B (p_{ib}^2 - 2p_{ib}p_i + p_i^2) + \frac{1}{B} \sum_{b=1}^B (2p_{ib}p_i - 2p_i^2) \right].$$

Recalling that we defined

$$p_i = \frac{1}{B} \sum_{b=1}^B p_{ib},$$

show that the second summation in the brackets is zero.

Exercise 4. Establish the identities in (13.12) and (13.13).

Exercise 5. Show that the expressions in (13.15) correctly represent the values of D_{\max} . [Hint: Work with the definitions in (13.11) and (13.12). For $D < 0$ set $p_{A_1B_1}$ or $p_{A_2B_2} = 0$, and note the constraints on $p_{A_1}p_{B_1}$ and $p_{A_2}p_{B_2}$, given the requirement that probabilities are always positive. Use a similar approach for $D > 0$.]

Exercise 6. Suppose that the proportions of haplotypes in a population are reflected by the proportions diagrammed in Fig. 13.3B (e.g., $p_{A_1B_1} = (4/10)$; $p_{A_1} = 5/10$). To confirm that the genealogy of mutations shown in Fig. 13.3A produces LD, use the data to compute $|D'|$ for loci A, B and $|D'|$ for loci B, C .

Exercise 7. For the Wright-Fisher Markov chain in Section 13.7.2, show that the probability that allele A becomes fixed is its initial relative frequency. [Hint: define

$$\pi_i = \mathbb{P}(X \text{ reaches } 2N \text{ before reaching } 0 | X_0 = i),$$

with $\pi_0 = 0, \pi_{2N} = 1$. By conditioning on the value of X_1 , justify the following equation satisfied by the π_i :

$$\pi_i = \sum_{j=0}^{2N} p_{ij} \pi_j$$

and verify that $\pi_j = j/(2N)$ solves the equation.]

Exercise 8. For the Wright-Fisher Markov chain in Section 13.7.2, find the variance of X_n , and use this to verify (13.21).

Exercise 9. The nucleotide diversity Π_n among a set of n aligned sequences was introduced in Section 13.8.2. It is defined by

$$\Pi_n = \frac{2}{n(n-1)} \sum_{i < j} d_{ij},$$

where d_{ij} is the number of segregating sites between sequences i and j . (This is the Hamming distance between the sequences.) Show that under the infinitely many sites model of mutation, the expected value of the nucleotide diversity is θ .

Exercise 10. In Section 13.8.3 we found a formula for $\mathbb{E}T_{\text{MRCA}}$ for a sample of n genes. Calculate the variance of T_{MRCA} and find its value for large samples.

Exercise 11. Calculate the mean and variance of the tree length L_n in (13.31) of a coalescent tree of size n .

Exercise 12. Watterson's estimator θ_W of θ was defined in (13.34).

- Calculate the variance of θ_W .
- Show that it decays at a rate proportional to $1/\log n$ in large samples, and comment on the practical implications of this result.

Exercise 13. R can be used to simulate observations having the distribution of θ_W .

- Write a function to simulate an observation ℓ having the distribution of L_n in (13.31). Exponential random variables T_2, \dots, T_n can be generated using `rexp`.
- Given the value of ℓ , generate a Poisson variable s having mean $\theta\ell/2$. Poisson random variables can be generated using `rpois`. Given the value of s , calculate an observation from θ_W from (13.34).
- For $\theta = 1, 5, 25$ and $n = 10, 25, 100$ simulate observations from the distribution of θ_W , and compare the results.

Exercise 14. This exercise focuses on the rejection algorithm for simulating from the posterior distribution of mutation rate and coalescence times described at the end of Section 13.8.4. It builds on the method developed in Exercise 13. For definiteness, assume that $\pi(\theta)$ is a uniform density over some range, so prior observations for θ can be generated using `runif`.

- Implement the rejection algorithm in R.
- The quantity h in Step 2 of the rejection algorithm can be replaced by

$$h = \frac{e^{-\theta L/2}(\theta L/2)^k/k!}{e^{-k}k^k/k!} = e^{k-\theta L/2}(\theta L/2k)^k,$$

resulting in a faster algorithm. Verify this by modifying your function in a.

- Given that $k = 5$, generate 1000 observations from the posterior distribution of θ for samples of size $n = 10$, and plot the estimated posterior density. The function `density` is useful for this. Explore the effects of different prior distributions on the posterior.
- How could you use the algorithm to generate observations from the posterior distribution of the time to the most recent common ancestor?

Comparative Genomics

Computational biology provides insights into the nature of genomes and organisms, and provides tools for understanding how an organism's characters or phenotypes are determined by its genome sequence. In prior chapters, we presented a number of computational methods addressing a variety of specific biological questions. In this concluding chapter, we indicate in more detail how these tools can be employed in the context of complete genomes. Computational analysis of genome sequence data has transformed the approach to answering biological questions because now they can be formulated in the context of all genes operating as a coordinated system. This more integrated approach complements the reductionist approach of traditional molecular biology.

Comparisons can be performed within genomes and between genomes. *Within-genome* comparisons focus on the genome of organism X and scan it from beginning to end, analyzing variations in base composition, k -tuple frequency, gene density, and numbers and kinds of transposable elements, and identifying any duplicated regions. *Between-genome* comparisons employ closely related organisms (e.g., to help identify conserved genes, gene organizations, and control elements) or more distant organisms (to identify genes that are restricted to particular clades of a phylogenetic tree). Such data can help trace evolutionary trajectories of organisms.

What properties can be used to describe genomes? We start with compositional measures stated in terms of k -word content. As indicated in Chapter 2, even simple measures such as this can prove very informative. Second, we examine the fraction of the genome represented by transposable elements. These elements may or may not represent a large proportion of the genome. Although they do not code for organism-specific proteins, the sequences of these elements can be used as tracers for defining the evolutionary histories of groups of like organisms. Third, we examine how sequence organization in chromosomes reflects duplication within genomes or evolutionary relationships between genomes. We have already touched on this question in Chapter 5. Fourth, we provide a brief introduction to the identification and char-

acterization of genes. Finally, we indicate how the predicted proteome can be functionally annotated and how proteomes of organisms can be compared.

14.1 Compositional Measures

Summaries and statistical descriptions of genomes are essential for comprehending genome content. The page that you are reading now can accommodate approximately 3500 characters. Approximately 500 to 1000 such pages would be required to print out the sequence of an “average” microbial genome, and about a million pages would be required to print out the sequence of the human genome. The human mind cannot grasp such quantities of data in this form. In this section, we illustrate what can be learned from simple k -word statistics, especially when these statistics are combined with other data. Even as simple a number as genome size, when combined with gene number (Table 14.1), imposes important constraints on genome composition. This is discussed in detail in Section 14.4.

k -word compositions of genomes measured as a function of position along the chromosome are not uniform. Even simple compositional measures can be very informative. The simplest measure is the base composition (corresponding to $k = 1$). Figure 14.1 shows distributions of GC content, transposable elements, and genes along human chromosome 17. Distributions of these features along chromosomes provide a statistical analog of a genetic map, and they are useful for understanding chromosome structure and function. The third panel from the top shows the %G+C as a function of position. The three regions where this quantity is the lowest correspond to regions where genes are particularly sparse. (Compare this with the bottom panel in Fig. 14.1). In the human genome, gene-rich regions typically have a higher %G+C than regions that are gene-poor.

Another statistic based on $k = 1$, and particularly useful for prokaryotic genomes, is the GC skew, defined in Chapter 2 as $(\#G - \#C)/(\#G + \#C)$. This statistic is computed for sliding windows of length w along the genome. Prokaryotic chromosomes are usually circular and usually have a single replication origin and a terminus located approximately 180° away from the origin. This means that in one half of the chromosome, the genomic DNA strand as written in the sequence file corresponds to DNA produced by *leading strand* DNA synthesis, while the other half of that strand is produced by **lagging strand** DNA synthesis. In many bacteria, the leading strand exhibits an excess of G relative to C, or a positive GC skew. The GC skew has been used to infer the origin of replication in sequenced genomes. An example is shown in Fig. 14.2 for *Borrelia burgdorferi*, which has a linear rather than circular chromosome (lower panel). The predicted replication origin would be near the 450kb position based upon the pattern of GC skew.

Genome regions having unusual shifts in one or more statistics may have particular biological interest. For example, *Yersinia pestis*, the causative or-

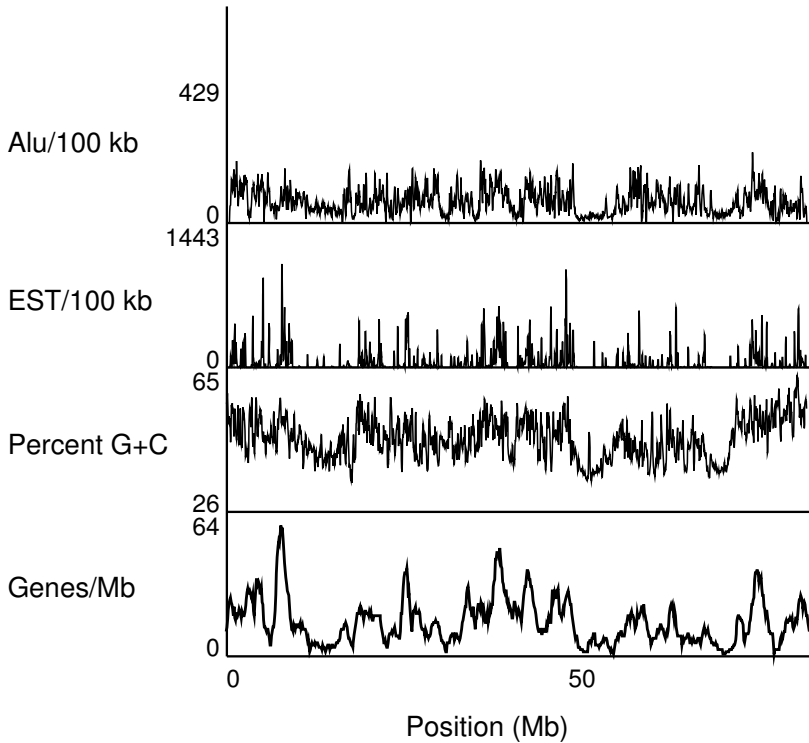


Fig. 14.1. Composition of human chromosome 17 (Hsa17). Various properties have been summarized for 100 kb intervals along the length of the chromosome. Reprinted, with permission, from Venter JC et al. (2001) *Science* 291:1304–1351. Copyright 2001 American Association for the Advancement of Science.

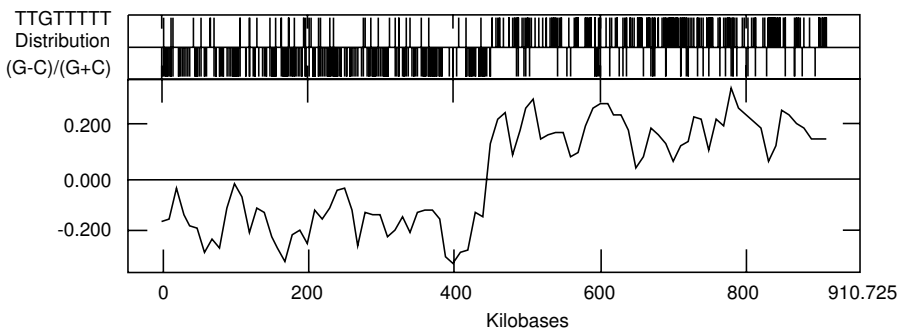


Fig. 14.2. GC skew for the *Borrelia burgdorferi* linear chromosome (lower panel) and distribution of eight-letter word TGT on “top” and “bottom” strands (upper two panels). GC skew measures the relative excess of G (compared with C) on a particular strand. Reprinted, with permission, from Fraser CM et al. (1997) *Nature* 390:580–586. Copyright 1997 Nature Publishing Group.

Table 14.1. Statistics describing genomes of common and model organisms. Human mitochondrial DNA is included as an example of an organellar genome.

Organism	Number of Genes	Genome Size (Mb)	Chromosomes ^a
(Human mtDNA)	37	0.016	$\sim 10^3 - 10^4$) ^b
<i>Mycoplasma genitalium</i> (bacterium)	517	0.58	1
<i>Escherichia coli</i> (bacterium)	4,288	4.64	1
<i>Saccharomyces cerevisiae</i> (baker's yeast)	6,000	12.05	16
<i>Caenorhabditis elegans</i> (nematode worm)	18,400	97	5+X
<i>Drosophila melanogaster</i> (fruit fly)	13,600	180	4
<i>Arabidopsis thaliana</i> (dicotyledonous plant)	25,500	125	5
<i>Fugu rubripes</i> (Pacific puffer fish)	31,000	365	22
<i>Homo sapiens</i> (mammal)	25,000	3,080	23
<i>Allium cepa</i> (onion)	NA	15,000	8

^a Haploid number N is reported for diploid organisms.

^b Organelle. Chromosome number is DNA copy number in somatic cells.

ganism for plague, or “Black Death,” is a potential biowarfare agent. *Yersinia pestis* is closely related to another *Yersinia* species, *Y. pseudotuberculosis*, which is not a blood-borne pathogen, but instead causes gastrointestinal disease. *Yersinia pestis* is thought to have evolved relatively recently from an ancestor shared with *Y. pseudotuberculosis* (perhaps during the last 20,000 years). Properties of the *Y. pestis* genome are summarized in Fig. 14.3. The inner circle in Fig. 14.3 is a plot of the GC skew. Notice that, for the most part, the genome is divided into approximate halves with the right half having a mostly positive GC skew and the left half having a mostly negative GC skew. There are three regions where the GC skew reverses sign for comparatively short portions of the genome, suggesting relatively recent inversion events.

Genes involved in *Y. pestis* pathogenicity or adaptation are marked in dark blue on the two outer circles of Fig. 14.3. It is known that genes conferring pathogenicity traits are often located in clusters, called pathogenicity islands, and that these may be acquired by horizontal transfer (from conjugative plasmids, bacteriophages, or other gene transfer mechanisms). DNA from such sources will not, in general, exhibit the same base composition or other statis-

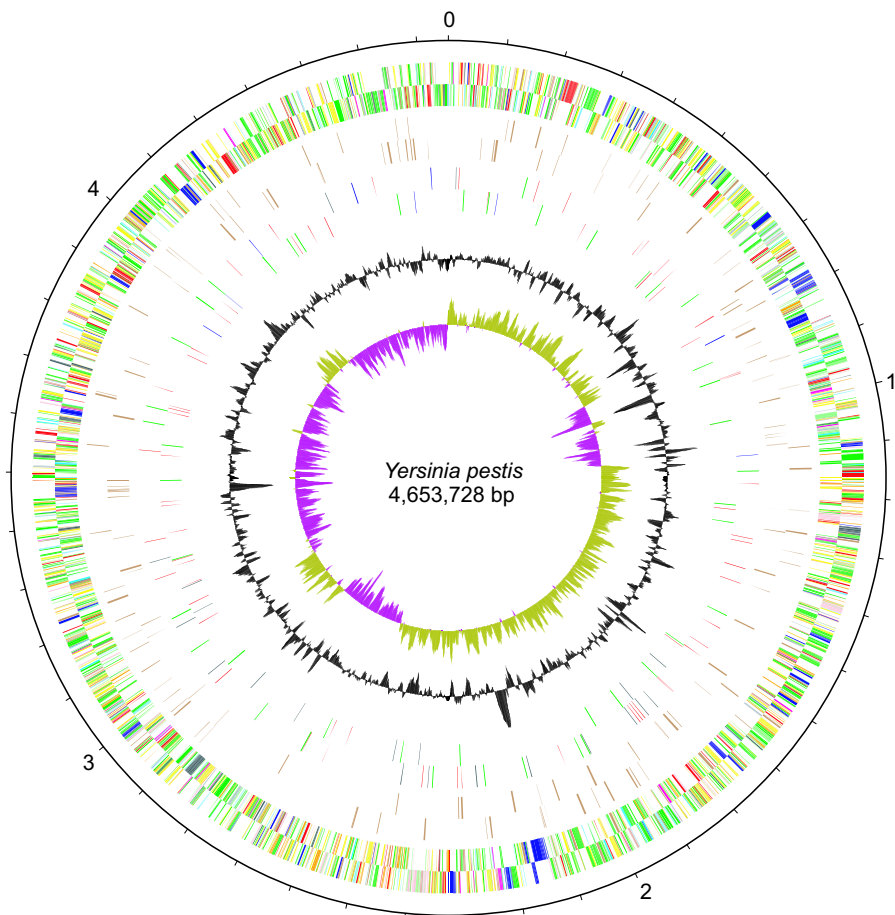


Fig. 14.3. [This figure also appears in the color insert.] Properties of the *Yersinia pestis* genome. Numbers on the outer circle denote coordinates in millions of bp clockwise of the map origin at 0. Two bands of short radial lines just inside the coordinate circle represent the genes on the two DNA strands. Dark blue lines denote genes related to pathogenicity and adaptation. The innermost circle represents GC skew, and the next one out (black) depicts variations in base composition relative to the mean. Reprinted, with permission, from Parkhill J et al. (2003) *Nature* 413:523–527. Copyright 2003 Nature Publishing Group.

tical properties as the genome that receives it. Notice that the pathogenicity genes at the 2.15 Mb position on the genome (between 5 and 6 o'clock) are clustered on one strand and that a peak of elevated %G+C appears at the same position (second circle from the inside). This type of pattern could result if a block of pathogenicity genes had been acquired from another organism.

Genomes can also be described by dinucleotide measures ($k = 2$; Sections 2.5 and 2.6). For example, probability distributions for dinucleotides can be organism-specific (Karlin et al., 1998). In human DNA, 5'-CG-3' (also known as CpG, where p stands for the phosphate residue) occurs with about 20% of the frequency anticipated for iid bases having the base composition of human DNA (IHGSC, 2001). However, there are regions of the genome where their frequency is closer to the predicted value. These regions are called CpG islands. In the human genome, there are approximately 29,000 CpG islands (a number similar to the predicted gene number), and most of them are less than 1800 bp long. There is a correlation between the density of CpG islands and the gene density, which is expected from prior experiments showing association between CpG islands and the 5' ends of vertebrate genes (see Strachan and Read, 2003, and references therein).

14.2 Transposable Elements

Studies of eukaryotic DNA by reassociation kinetics in the 1960s and 1970s showed that eukaryotic genomes contain different sequence components that vary by copy number. Some genomic sequences appear once or a few times (**unique sequence**), while others appear many times (**repeated sequences**). One class of repeated sequences is the telomeric repeats (the short repeated sequences found at the ends of chromosomes), which are important for chromosome stability. Another class of repeated DNA consists of multiple copies of one or more types of **transposable elements**. These can represent a substantial fraction of some eukaryotic genomes. For example, about 45% of the human genome is attributable to transposable elements. In contrast, the percentages of transposable element DNA in *Arabidopsis*, *Caenorhabditis elegans*, and *Drosophila* genomes are 10.5%, 6.5%, and 3.1%, respectively (IHGSC, 2001). For efficient computational analysis of genomes, it is necessary to know the number of transposable element classes, the number of elements in each class, and how the elements are distributed along the genome. These multiple copies can complicate DNA sequence assembly in whole-genome shotgun approaches. Highly repetitive sequences are usually “masked” (omitted) before applying exon prediction tools, or before making interspecies comparisons.

The types and copy numbers of transposable elements within the human genome are shown in Fig. 14.4. There are generally two categories of each type of transposon. One category encodes functions required for autonomous transposition, and the other category lacks one or more of these functions (i.e., elements of this category are defective). Transposition of defective elements

requires gene products supplied in *trans* from related elements. One type of transposon is the **LTR** transposon, where LTR stands for long terminal repeat, a characteristic of retroviruses. Its nonautonomous counterparts possess the LTRs, but they lack the reverse transcriptase. Another general type of element is the LINE (long interspersed nuclear element), which also uses reverse transcription to transpose but does not require or possess LTRs. Its nonautonomous counterparts are the **SINEs** (short interspersed nuclear elements.) The third type of element consists of DNA transposons (whose mechanism does not require reverse transcription) and their defective partners, which lack the transposase encoded by the autonomous counterpart.

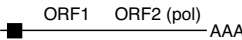

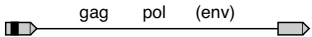
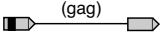
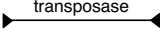

			Length	Copy number	Fraction of genome
LINEs	Autonomous		6-8 kb	850,000	21%
	Nonautonomous		100-300 bp		
Retrovirus-like elements	Autonomous		6-11 kb	450,000	8%
	Nonautonomous		1.5-3 kb		
DNA transposon fossils	Autonomous		2-3 kb	300,000	3%
	Nonautonomous		80-3000 bp		

Fig. 14.4. Types and numbers of transposable elements found in the human genome. Reprinted, with permission, from International Human Genome Sequencing Consortium (2001) *Nature* 409:860–921. Copyright 2001 Nature Publishing Group.

Also shown in Fig. 14.4 are the numbers of each type of element. Over a million copies of Alu elements (a type of SINE) are found within the human genome. They comprise about 10% of the human genome, and they appear on average once every three kb. They do not contribute to the protein encoded by those genes into which they are inserted. Since the average human gene extends over 27 kb, we would expect (on average) to find nine such elements in the noncoding regions of an average gene, provided Alu elements are targeted uniformly throughout the genome. (It should be obvious why Alu elements are absent from coding sequences.) In fact, targeting preferences to different parts of the genome are not the same (Alu elements seem to target AT-rich DNA more frequently), but the observed distribution of these elements is complicated by post-transpositional losses (deletion). There are more than 500,000 copies of LINE L1 elements in the human genome, suggesting that they appear on average once every 6 kb. Thus, we would also expect to find several L1 elements in regions the size of an average gene. L1 elements, like Alu sequences, also seem to preferentially target AT-rich DNA.

Most transposable elements in genomes are defective. If all were active, the mutational “load” associated with the large numbers of transposable elements found in many eukaryotes would probably be incompatible with species survival. Because most genomic transposable elements have been mutationally inactivated during the course of evolution, they are not under selection for function, and therefore they continue to accumulate additional mutations. If we align instances of any particular type of transposon, we can define a consensus sequence for that element (presumably corresponding to a functional element having no mutations). We can then distribute all elements into “bins” having differing levels of sequence divergence from this consensus. The degree of sequence divergence from the consensus is a measure of how long ago the elements in each bin were transpositionally inserted: higher levels of divergence correspond to more remote times. An example of this is shown in Fig. 14.5. Bins of each element type having large proportions of elements correspond to periods when transposition of that element was particularly active. It is evident that most Alu transpositional events were relatively recent in humans but that LINE L2 elements were primarily active in the remote past and are no longer transposing at a significant rate. The distribution of transposable element sequences as a function of sequence divergence provides a measure of the evolutionary trajectory of the organism. Even recently transposed elements can provide evolutionary information over shorter timescales. For example, Alu insertions can be used to analyze phylogenetic relationships among great apes (Salem et al., 2003).

14.3 Sequence Organization within Chromosomes

Even before the advent of genome sequencing, it was clear that DNA within a chromosome could be divided into distinguishable regions. The first division of DNA into types is its assignment to either **euchromatin** or **heterochromatin**. Recall that chromatin is composed of DNA with the bound histones and other chromosomal proteins. Euchromatin is more actively transcribed, less condensed during interphase, and exhibits cytological staining properties different from heterochromatin, which remains condensed during interphase and is relatively inactive transcriptionally. Heterochromatic regions are commonly found near the centromeres of chromosomes. Recently, heterochromatin has been defined operationally as that portion of the genome that cannot be readily cloned in high-capacity cloning vectors such as BACs (Adams et al., 2000). This latter property, thought to relate to the relatively large number of repeated sequences in heterochromatin, has slowed the completion of genomic sequencing in some organisms. Organisms can differ significantly in the amounts of heterochromatin that they contain. For example, heterochromatin represents about 6.5% of the human genome but approximately 33% of the *Drosophila melanogaster* genome. Draft sequences at the time of initial publication may exclude large proportions of the heterochromatic regions. In

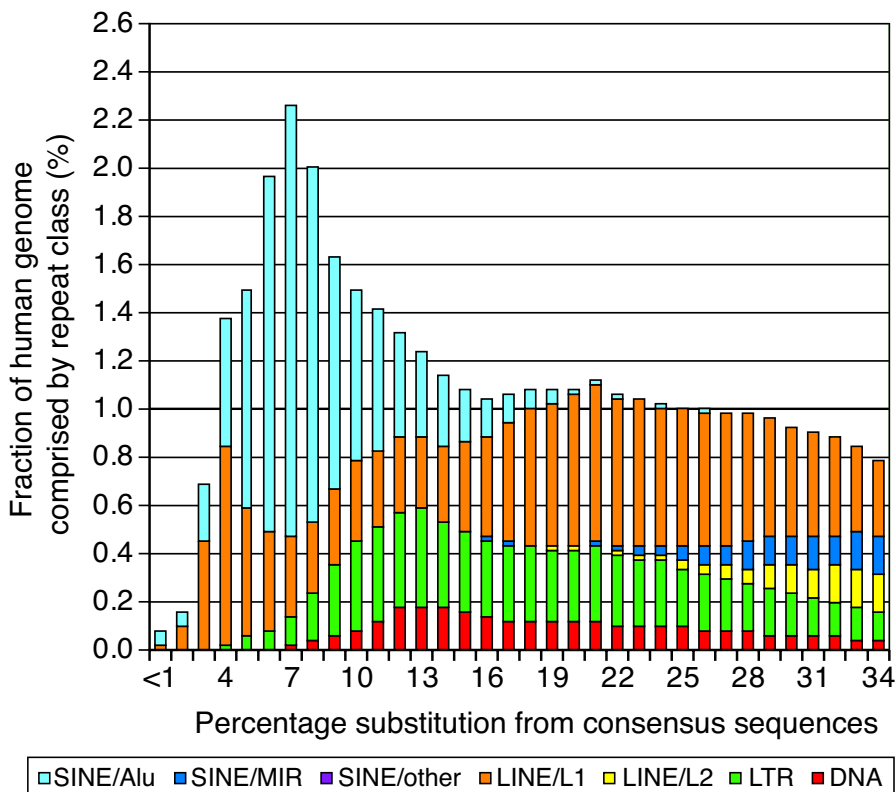


Fig. 14.5. [This figure also appears in the color insert.] Abundance and sequence divergence of different classes of human transposable elements. Light blue: SINE Alu I; dark blue: SINE Mir; green: LTR transposons; orange, LINE L1; yellow: LINE L2; red: DNA transposons. Higher percentages of substitution from the consensus sequence correspond to more ancient copies of elements. The higher proportion of Alu elements at lesser amounts of substitution indicates more recent transpositional activity among this class of elements, while the absence of LINE L2 elements at low levels of substitution indicates that L2 elements were not transposing recently. Reprinted, with permission, from International Human Genome Sequencing Consortium (2001) *Nature* 409:860–921. Copyright 2001 Nature Publishing Group.

the case of *Drosophila*, this amounted to about one-third of the total genome (Adams et al, 2000).

Genetic mapping studies allow comparisons within and between genomes at a map resolution determined by marker or gene densities. Map comparisons can reveal genome segments that have been duplicated within genomes or segments whose genetic organization has been conserved between genomes. Genome sequences increase map resolution by several orders of magnitude, which allows for much more detailed comparisons within and between genomes.

If two species are very closely related, we might expect that their genetic maps would be very similar. In other words, the gene orders and relative locations may be similar, at least over short distances. We already saw an example of this in Fig. 5.1 of Chapter 5, where contents of a mouse chromosome and human chromosomes were compared. We also saw in that chapter how alterations in gene orders could serve as measures of the evolutionary distance between organisms.

14.3.1 Conservation of Synteny and Segmental Duplication

As indicated in Section 1.3.2, two or more genes are said to be **syntenic** if they reside on the same chromosome. A set of genes g_1, \dots, g_n that is syntenic in organism A and also syntenic in organism B represents a **conserved synteny**. Note that the chromosomes in the respective organisms may not be related to each other in a simple way. A group of genes that display conserved synteny *and* that appear in the same order and relative map positions in the two genomes constitutes a **conserved segment** (also called a *conserved linkage* or *syntenic segment*; see Fig. 1.4B). As was illustrated in Fig. 5.1, there are multiple instances of conserved segments shared by the human and mouse genomes, even though these two lineages diverged from each other more than 83 million years ago. The existence of conserved segments indicates that analysis and annotation of such segments in one genome can guide analysis in a related genome. For example, the human genome sequence provided a useful framework for generating a high-resolution physical map for the mouse genome (Gregory et al., 2002). This was done by matching end-sequence reads from inserts in a mouse BAC library with the assembled human genome sequence. The tiling could be checked by comparing fingerprints (patterns of insert restriction fragments) of overlapping BACs.

Once conserved segments between genomes have been identified by approaches to be described below, the rearrangements that have occurred since the two organisms diverged from the common ancestor can be inferred. A particularly simple example is the X chromosome in humans (HsaX) and mice (MmuX). The plot in Fig. 5.3 showed that runs of contiguous loci in MmuX have homologs in HsaX. These runs in the two organisms could be related by a series of reversals. Other mouse-human chromosome comparisons (Fig. 14.6) show much more complicated relationships involving many chromosomes. For example, Hsa3 has conserved segments corresponding to six different mouse chromosomes. Of course, both mouse and human contain DNA unique to each organism (white regions, Fig. 14.6). Moreover, each organism contains organism-specific transposable elements.

Within-genome comparisons can reveal large and small genomic regions that appear repeatedly, even though they are not transposable elements. **Segmental duplication** refers to duplicated regions that encompass only a fraction of the genome. The *Arabidopsis* genome provides a striking example of

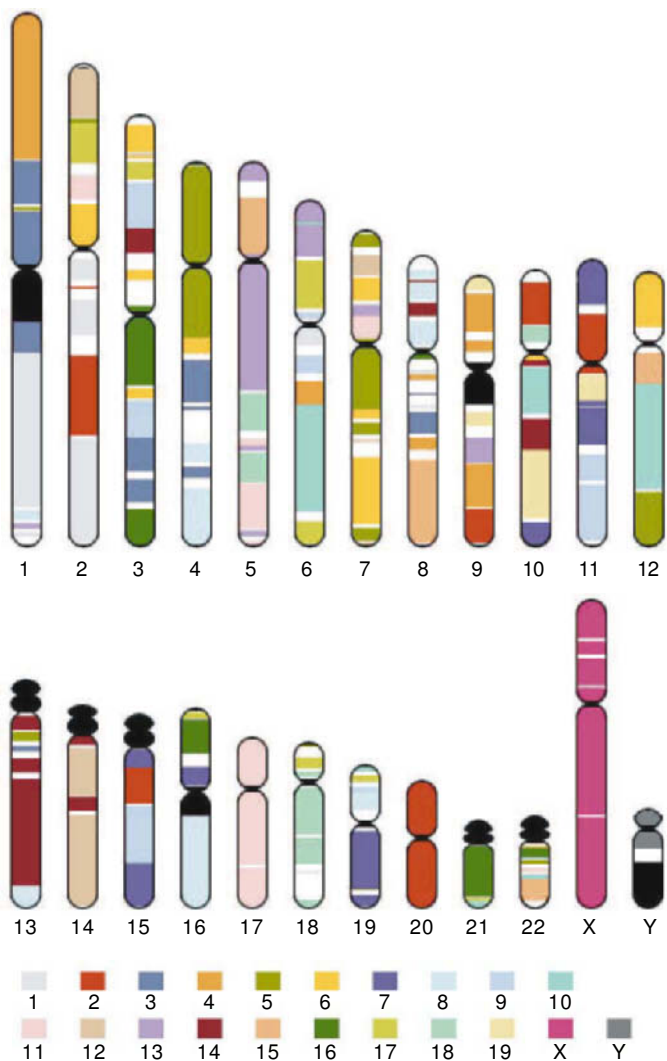


Fig. 14.6. [This figure also appears in the color insert.] Regions containing conserved segments in human and mouse chromosomes. Ideograms of human chromosomes are shown at the top. Color coding indicating the identity of mouse chromosomes (bottom) shows which regions of the human chromosomes are similar to the respective mouse chromosomes. Note that nearly all of the small human chromosome Hsa20 corresponds to a portion of the large mouse chromosome Mmu2. Portions corresponding to Mmu2 are also found on Hsa2, Hsa9, Hsa10, Hsa11, and Hsa15. Reprinted, with permission, from International Human Genome Sequencing Consortium (2001) *Nature* 409:860–921. Copyright 2001 Nature Publishing Group.

within-genome segmental duplication (Fig. 14.7). The obvious extensive duplication of sequences between chromosomes accounts for about 60-80% of the entire nuclear genome (Arabidopsis Genome Initiative, 2000; Simillion et al., 2002). In contrast, only about 5.3% of the human genome is segmentally duplicated (IHGSC, 2004).

14.3.2 Identifying Conserved Segments and Segmental Duplications

What statistical criteria are used to define instances of segmental duplication or conserved sequence segments? These criteria can be stated for different levels of resolution (Fig. 14.8). At the first level, we might assert that two regions are segmentally duplicated (or represent conserved segments in two different organisms) if they share the same set of protein-coding genes in the same order (Fig. 14.8A). Such sets of genes are called *collinear gene clusters*. Mural et al. (2002) defined collinear gene clusters as clusters having at least three genes meeting this criterion.

At higher resolution, we can perform sequence alignments of genomic segments with other segments in the same or a different genome. A segmental duplication (or conserved segment) of segment X would be identified if other segments are found having lengths and levels of sequence identity above arbitrary thresholds. (Clearly, transposable element sequences should be ignored before the comparison is performed.) There are two different implementations of this second approach. We could set a low sequence identity threshold, but demand alignment over larger regions, as was done in one of the analyses of the *Arabidopsis* genome. In that case, segments longer than 1000 bp and having more than 50% sequence identity were sought (Fig. 14.8B). Or we could use a set of smaller “anchor” sequences between regions being compared. These are relatively short (e.g., 200 bp) DNA segments (not necessarily within genes) having higher levels of sequence conservation (e.g., more than 85% sequence identity). Regions corresponding to segmental duplication or conserved segments would be recognized as “runs” of corresponding anchor sequences in the same order in the two regions being compared (Fig. 14.8C).

A third way of identifying segmental duplication (applicable for comparisons within but not between genomes) relies on statistics associated with sequence reads generated during whole-genome shotgun assembly. This is illustrated in Fig. 14.9. Recall that whole-genome shotgun assembly employs reads from the ends of large- and small-insert clones. These reads are relatively short, usually around 500-800 bp. The average coverage (read “depth”) is sufficiently large (typically greater than 5) to ensure collection of sequence data from almost all of the genome. Now consider a DNA segment Z that became duplicated sometime in the past (Fig. 14.9A). Since that time, both copies, Z1 and Z2, have independently accumulated mutations, so that the reads pooled from these two regions will have lesser amounts of sequence identity than would be the case for unique sequences. The method starts with

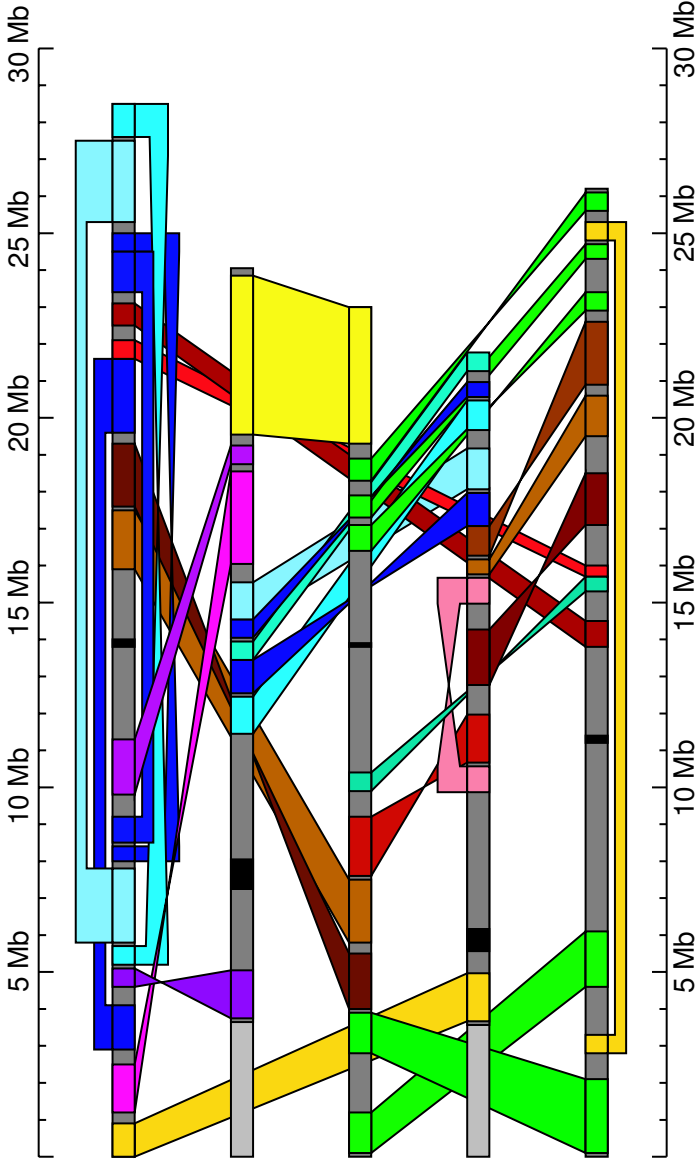


Fig. 14.7. [This figure also appears in the color insert.] Segmental duplications within the *Arabidopsis thaliana* genome. Black boxes denote the centromeres in each of the five chromosomes. Color-coded connecting ribbons indicate similar duplicated regions between and within chromosomes. Ribbon edges that cross indicate inversion of gene order in one segment compared with the other. Reprinted, with permission, from The Arabidopsis Genome Initiative (2000) *Nature* 408:796–815. Copyright 2000 Nature Publishing Group.

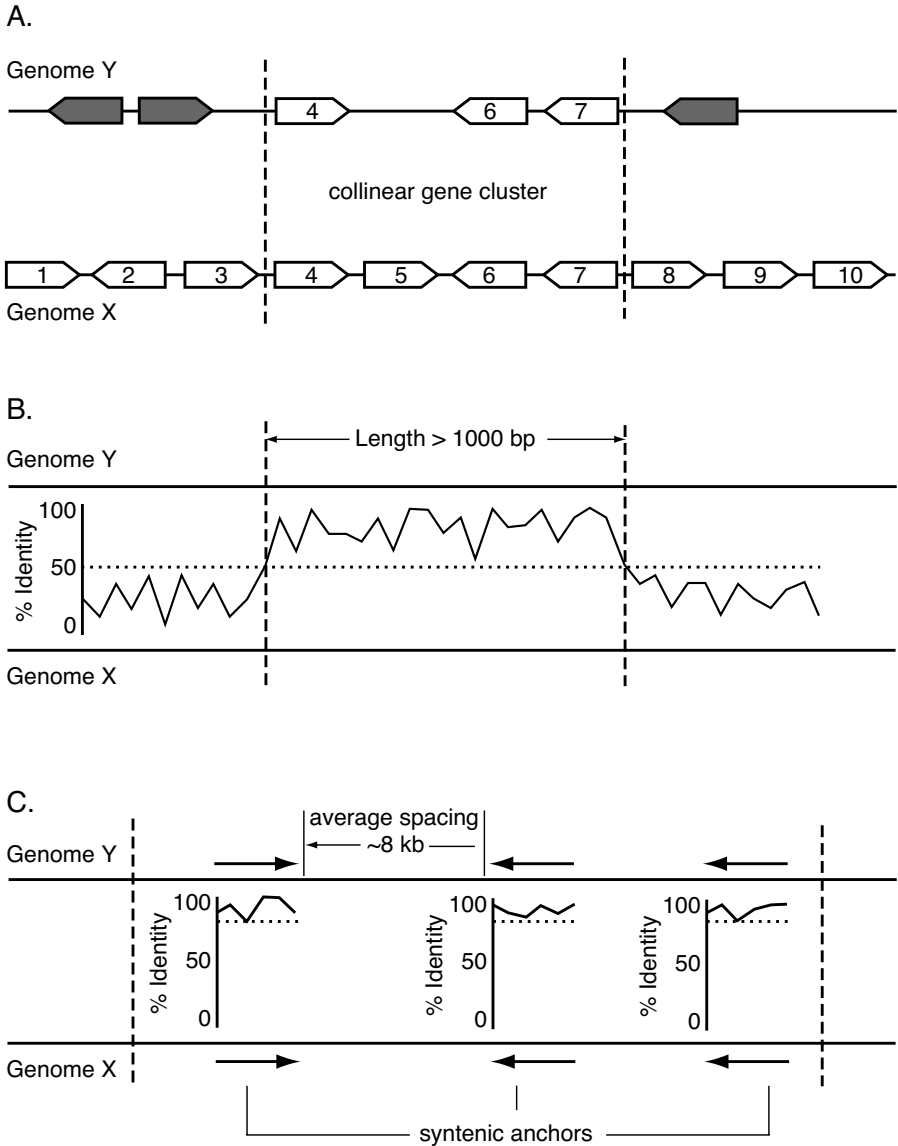


Fig. 14.8. Criteria for identifying conserved sequence segments in two genomes, X and Y. Conserved segments are regions between the vertical dashed lines. Panel A: Conserved segment defined by collinear gene clusters. Shaded genes are unrelated to unshaded ones. Panel B: Conserved segment identified by alignment with low identity threshold (dotted horizontal line) and minimum alignment length. Panel C: Use of anchor sequences having high identity thresholds and identified as reciprocal “best hits” in genomes X and Y. For the human genome at 88% identity, typical lengths of syntenic anchors are approximately 200 bp, and typical spacings are about 8 kb (Mural et al., 2002).

the assembled genomic sequence and then aligns with it all reads that exceed a specified threshold of sequence identity (e.g., 95%). Because sequence reads from *both* duplicated regions can align within *each* region, there is an increase in the coverage (read depth) within the duplicated region (twofold in the illustration). Moreover, the percentage identity of reads covering these regions is lower than for unique sequence regions. Regions of the genome having increased read depth and decreased levels of sequence identity may be identified as segmental duplications, as shown in Fig. 14.9B (Bailey et al., 2002).

14.3.3 Genome Evolution by Whole-Genome Duplication

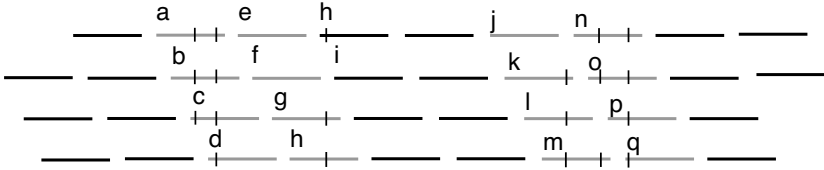
As indicated earlier (Section 14.3.1 and Chapter 5), one way that genomes evolve is by duplication of genome segments with subsequent genome rearrangement and deletion. In 1970, Susumu Ohno proposed an alternative model. He suggested that vertebrate genomes have evolved by *whole-genome duplications* of ancestral genomes. Such processes have occurred in higher plants (angiosperms): extant variants having two genome equivalents (tetraploids) and three genome equivalents (hexaploids) are well-known. For *Arabidopsis* and baker's yeast (*Saccharomyces cerevisiae*), evolution of contemporary genomes as a result of one or more earlier whole-genome duplication events is now well-supported. These two alternative mechanisms for genome evolution—*independent segmental duplication with rearrangement or whole-genome duplication*—have been epitomized as the “Slow Shuffle” or the “Big Bang,” respectively (Smith et al., 1999).

Whole-genome duplication can provide raw material for evolution because it produces paralogs of every gene in the genome. One member of each paralogous pair is available to supply the necessary ancestral gene function, while the other may be mutationally inactivated, deleted, or may evolve new functions (usually related to the original function). The presence of numerous duplicated genes within a genome is one indicator of an earlier, whole-genome

Fig. 14.9(Following page). Identifying duplicated regions from the number of reads generated during whole-genome shotgun sequencing. Panel A: Effects of segmental duplication on the number of reads and degree of sequence conservation between duplicated regions. In the assembled sequence, the heavy line represents unique sequences and open boxes represent duplicated sequences. Vertical lines represent positions that differ between duplicated sequences. Some of the sequence reads in the assembled sequence that are wholly or partially composed of the duplicated elements are indicated in grey and labeled. When sequence reads are aligned with the assembled sequence, reads from the second element also align with the first and vice versa. Dashed boxes enclose sequence reads from aligned duplicated sequences. Panel B: Illustration of how read “depth” and dissimilarity increase within duplicated regions in a portion of the human genome. Panel B excerpted, with permission, from Bailey JA et al. (2002) *Science* 297:1003–1007. Copyright 2002 American Association for the Advancement of Science.

A.

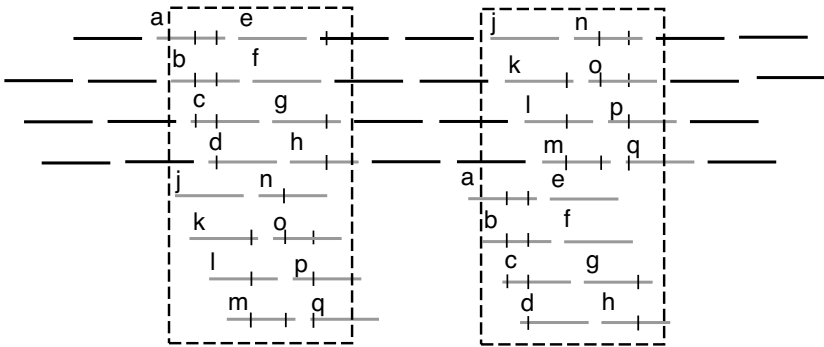
Reads producing assembled sequence



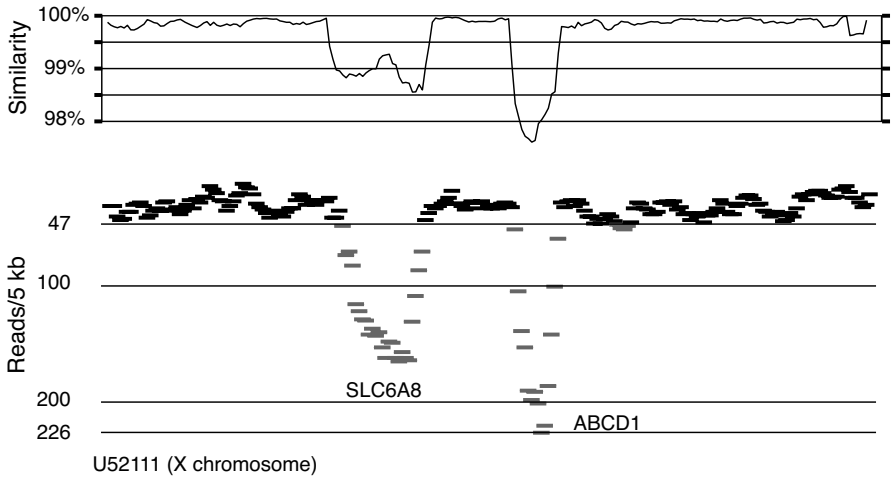
Assembled sequence



Reads aligned with assembled sequences



B.



duplication. However, over time, deletions, translocations, and inversions may erase some of the evidence of the original duplication event, making it hard to detect from within-genome sequence comparison. Although the baker's yeast genome is the result of a whole-genome duplication (see below), its genome is only 12% larger than the unduplicated genome of a comparable yeast species, *Kluyveromyces waltii*, and the number of predicted genes is only 10% greater.

There are several ways to infer whole-genome duplication events that have led to a contemporary genome. An obvious potential indicator is the presence of *duplicate genes*. However, these can also arise from independent duplication events, and if a sufficient proportion of paralogs has been lost, it may be difficult to distinguish the results of whole-genome duplication from the consequences of many smaller independent duplications. Sometimes sequence comparisons allow estimates of the time elapsed since the paralogs were formed, and many paralog pairs dated to the same historical time interval may indicate whole-genome duplication rather than a set of independent segmental duplication events. *Patterns of collinear gene clusters or anchor sequences*, described above, may also be indicators of whole-genome duplication. Collinear gene clusters can be used to detect whole-genome duplications even if the fraction of paralogous genes is small. We illustrate this for two different methods using either between-genome or within-genome comparisons.

Between-genome comparisons for identifying genome duplications leading to species D requires a closely-related species A whose genome is unduplicated and not extensively rearranged. A simplified illustration of this process is shown in Fig. 14.10. Actually, a particular organism whose genome has been duplicated might lie at any stage in the overall path illustrated, and the comparison shown in Fig. 14.10E will apply to syntenic segments rather than the whole genome because of translocations and inversions (not illustrated) that also occur. The processes of mutation and deletion are also concurrent rather than strictly sequential, as shown in this simplified diagram.

We can simulate the overall process by using standard playing cards. Select one suit (\diamond for example), and shuffle cards of that suit to represent the gene order of an unduplicated genome. Then lay out \spadesuit and \heartsuit in two adjacent horizontal rows, each with cards in the same order as the \diamond cards. The \spadesuit and \heartsuit together represent the duplicated genome. To simulate the result when only one member of each paralog remains after a series of deletion events following the initial duplication, sequentially "delete" either a \spadesuit or a \heartsuit of each card type, going from left to right. The suit to be deleted for each card type is chosen randomly by a coin toss. One realization of this procedure is shown below:

\diamond 6 7 5 4 Q K A 2 3 9 10 J 8	(ancestral order)
\spadesuit 5 4 A 2 9 J	(descendant after duplication and deletion)
\heartsuit 6 7 Q K 3 10 8	

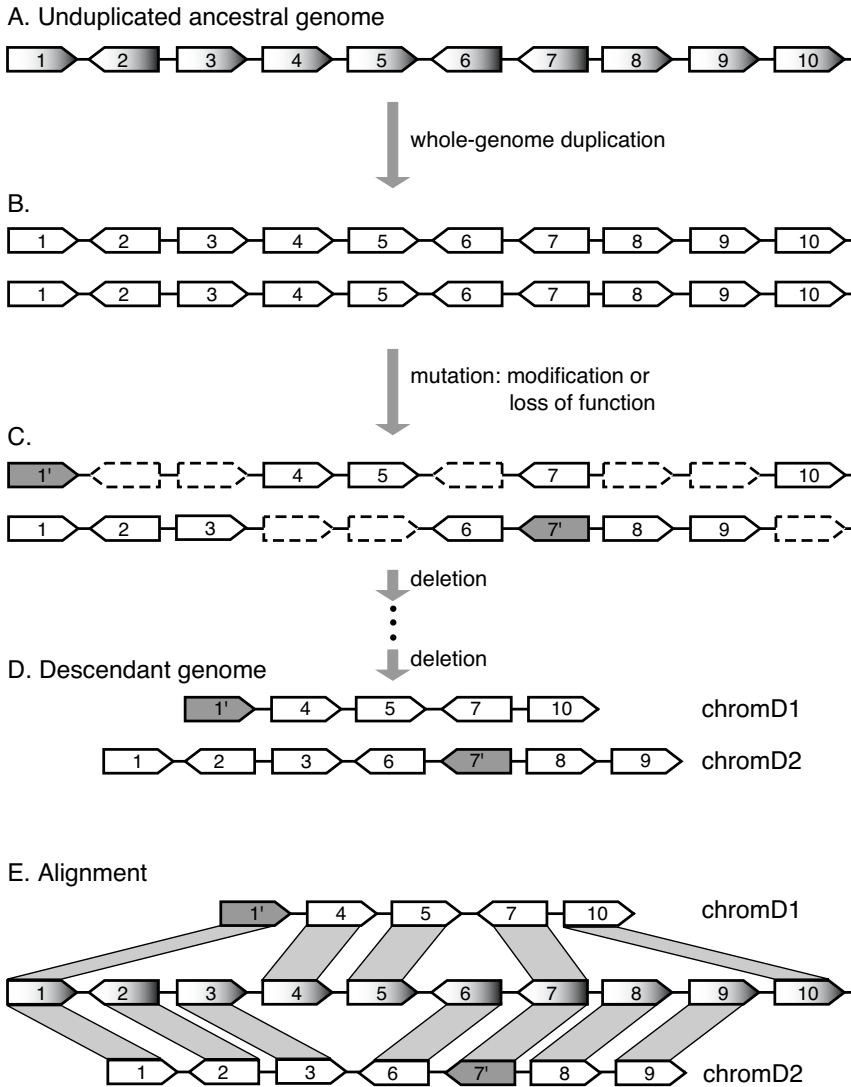


Fig. 14.10. Identifying whole-genome duplication by between-genome comparison. The unduplicated genome is shown in panel A. Genes are represented as pentagons with a gradient of shading. After whole-genome duplication (panel B), two copies of every gene are produced (unshaded pentagons). Genes can mutate (panel C), and these mutations may destroy functions (indicated by broken bounding lines) or alter them (shaded genes labeled with primes). Deletions may occur (panel D), subject to the constraint that essential genes be retained. Alignment of chromosomes D1 and D2 with the unduplicated genome of another descendant of the shared common ancestor (panel E) supports the whole-genome duplication because nonparalogous genes in the two chromosomes appear interleaved and in the same order as genes in the unduplicated chromosome.

There is no obvious way to determine from ♠ and ♥ cards alone that they resulted from a duplication event since each “gene” is a single copy. But observe the result of alignment with the unduplicated ◇ genome:

♠	--	5	4	--	A	2	9	--	J	--			
◇	6	7	5	4	Q	K	A	2	3	9	10	J	8
♥	6	7	--		Q	K	--	3	--	10	--	8	

The “genes” shared by ♠ and ◇ are in corresponding gene order, as is also true for genes shared by ♥ and ◇. Furthermore, when aligned with the unduplicated ◇ genome, ♥ and ♠ genes are *interleaving* (i.e., ♠ appears where ♥ is missing, and vice versa). This illustrates two results expected from genome duplication using the model specified in Fig. 14.10: conserved gene order, and interleaving of genes within corresponding duplicate segments (Fig. 14.10E). These observations are unlikely by the alternative (slow shuffle) model. (Try aligning the original ◇ sequence with ♣ that have been shuffled from the same starting sequence and then cut into two rows of six and seven cards.)

Whole genome duplications in yeast have been recognized from the conserved gene order or interleaving of genes in duplicate regions (Wong et al., 2002). This approach verified that the *Saccharomyces cerevisiae* genome is a product of whole-genome duplication, as evidenced by comparison with the genomes of yeasts *Kluyveromyces waltii* (Kellis et al., 2004) or *Ashbya gossypii* (Dietrich et al., 2004). The duplicate mapping and interleaving of genes near the centromeres of *S. cerevisiae* chromosomes Scer12 and Scer10 relative to *K. waltii* chromosome Kwal5 is shown in Fig 14.11. Note that for *S. cerevisiae* there are 16 chromosomes, compared with 8 from *K. waltii*. This is consistent with a whole-genome duplication event, which will double the number of centromeres. The *S. cerevisiae* gene set includes about 88% of the *K. waltii* genome and contains about 457 paralogous gene pairs (8% of the *S. cerevisiae* gene set) (Kellis et al., 2004). Because of genome rearrangements and other effects in the independent lineages leading to *K. waltii* and *S. cerevisiae* from their common ancestor, gene orders and orientations are not conserved over entire chromosomes, but rather are limited to conserved segments spanning 27 genes on average.

The second method, *within-genome comparison*, has been used to identify multiple genome duplications even when substantial genome rearrangements have occurred. The method (Fig 14.12) is illustrated for collinear gene clusters (see Fig 14.8A) whose homologous genes have been detected by using BLAST. Two whole-genome duplications are depicted. Although this should produce four collinear gene clusters, cluster C has been lost as a result of subsequent deletion events. Thus, the multiplication level (number of duplicate copies) for this genome region is three rather than four. Shared homologous genes define a collinear gene cluster contained within A' and B' and another cluster contained within B' and D'. A' and D' do not share any homologous genes,

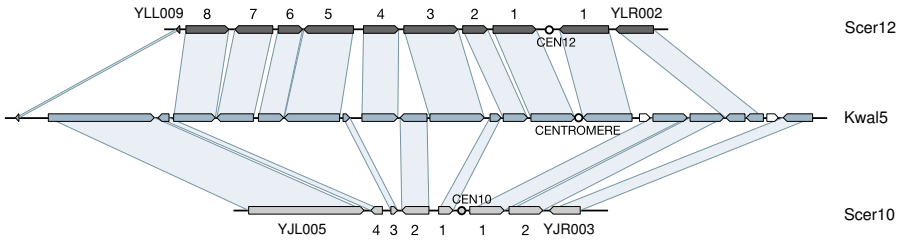


Fig. 14.11. Example of duplicate mapping of centromeric regions from *S. cerevisiae* chromosomes Scer12 and Scer10 to the same region near the centromere of *K. waltii* chromosome Kwal5. Although neither *S. cerevisiae* chromosome has all of the genes found in this region of Kwal5, the gene orders are the same as in Kwal5, and the genes of *S. cerevisiae* interleave to tile almost the entire portion of Kwal5 shown. Excerpted, with permission, from Kellis M et al. (2004) *Nature* 428:617–624. Copyright 2004 ES Lander.

but *transitive homology* can be inferred: since region A' is homologous to B' and region B' is homologous to D' , we conclude that A' is homologous to D' . This type of analysis gives a better assessment of multiplication levels.

This method has been used to determine the number of whole-genome duplications in the history of *Arabidopsis* (Simillion et al., 2002). Multiplication levels for direct homologies of collinear gene clusters measured along the genome mostly ranged from two to four. When transitive homologies were included, many regions with multiplication levels of five to eight were observed. This suggested that three whole-genome duplication events had occurred ($5 > 2^2$ and $8 = 2^3$). The fraction of synonymous substitutions per synonymous site, K_s , was used to estimate dates of 75, 160, and 220 million years ago for these duplication events.

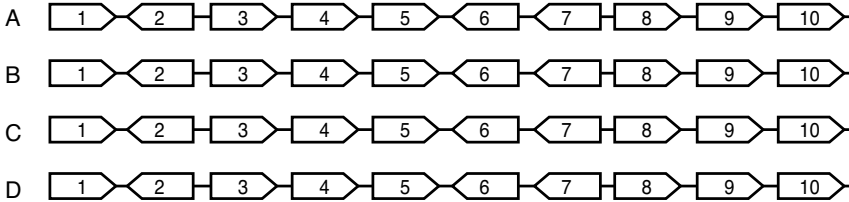
Other methods for identifying and dating genome duplication events are available (Van de Peer, 2004). Some of these methods employ clusters of homologous genes that appear together at different chromosomal locations, but not necessarily in the same order. The statistical significance of these homologous clusters can be estimated by randomly shuffling the gene map and identifying any homologous clusters that appear. For each pair of homologs, additional homologs closer than a specified distance (measured either by window size or by the number of unduplicated genes that separate them) constitute randomly-generated gene clusters. The probability of observing these gene clusters is approximated after 1000 such simulations. For recent research on this and related statistical problems, see Durand and Sankoff (2003).

Analysis of the human genome by using clusters of paralogous genes, or paralogons, indicates that at least one whole-genome duplication has occurred in the chordate lineage (McLysaght et al., 2002). The estimated date for this duplication is 350–650 million years ago. As additional complete genome sequences from throughout the phylogenetic tree become available, it will be

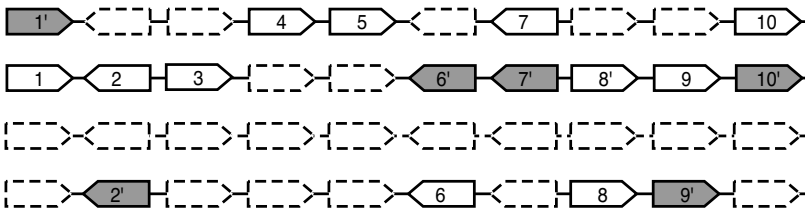
Unduplicated ancestral genome region



↓ whole-genome duplication
 ↓ whole-genome duplication



↓ mutation: modification or loss of function



↓ deletion
 ⋮ translocation
 ↓ recombination

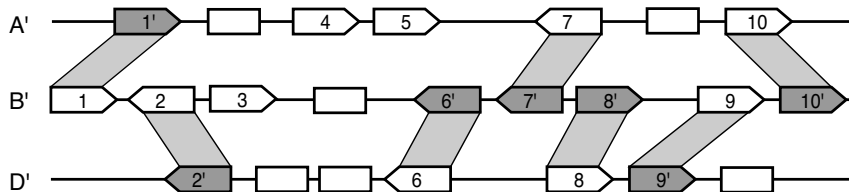


Fig. 14.12. Identifying whole-genome duplication by within-genome comparison. Open boxes represent genes not involved in this duplication event. Other graphical conventions are the same as in Fig. 14.10. A segment of the unduplicated genome is shown at the top. Two rounds of whole-genome duplication produce four copies A, B, C, and D. After subsequent deletion, mutation, translocation or other events, collinear gene clusters within (A', B') and (B', D') remain, indicating a multiplication level of three for this region. Homology of D' with A' is inferred even though they share no homologous genes.

possible to refine hypotheses about genome evolution by whole-genome duplication.

14.4 Gene Content

Two major tasks to accomplish after assembly of a complete genome sequence are identifying gene sequences and assigning gene functions. Functional assignment is discussed in Section 14.5. Here we focus on identifying genes and their associated regulatory sequences. Three types of information employed in gene finding are signals (e.g., splice sites), sequence content (e.g., *k*-word frequencies), and similarities with cDNAs or ESTs (Stormo, 2000). The first two types of information depend upon the local sequence context, while the latter type employs comparison with database entries.

The different categories of genome sequence are shown schematically in Fig. 14.13. Once transposable elements and other repeated sequences have been identified and suitably masked, the remaining single-copy genomic sequence can be further characterized as either **coding sequence** or **noncoding sequence**. Information represented in coding sequences will eventually appear in gene products either as RNA (e.g., tRNA) or proteins. Coding regions represent only a small proportion of the genomes of complex eukaryotes such as plants and animals (Table 14.2). The percentage of the DNA that codes for genes indicates the magnitude of the gene identification problem. The yeast *S. cerevisiae* genome, for example, contains 70% coding sequences, which means that the signal-to-noise ratio is high. The human genome, in contrast, contains only about 1.2% protein coding sequences, and just 5% of each protein-coding gene actually codes for amino acid residues (the signal-to-noise ratio is low). Such genome properties will affect which experimental and computational approaches are appropriate.

The classical experimental approach to gene finding is to generate mutations in an organism, characterize the resulting phenotypes, genetically map the locations of the mutations in the genome, and then identify the altered gene products biochemically. This traditional method is arduous and time-consuming. Genetic engineering methods for systematically targeting particular genes for mutation (i.e., gene “knockout” methods) are available and practical for some organisms. For example, 96% of the open reading frames of baker’s yeast (*S. cerevisiae*) were deleted one at a time, and effects upon cell morphology and growth were measured under a variety of conditions (Giaever et al., 2002). However, knockout technologies may not be available for all organisms, or in the case of humans are not applicable for ethical reasons. For such cases, and in general as a preliminary, it is usual and convenient to employ computational methods for initial characterization of sequenced genomes.

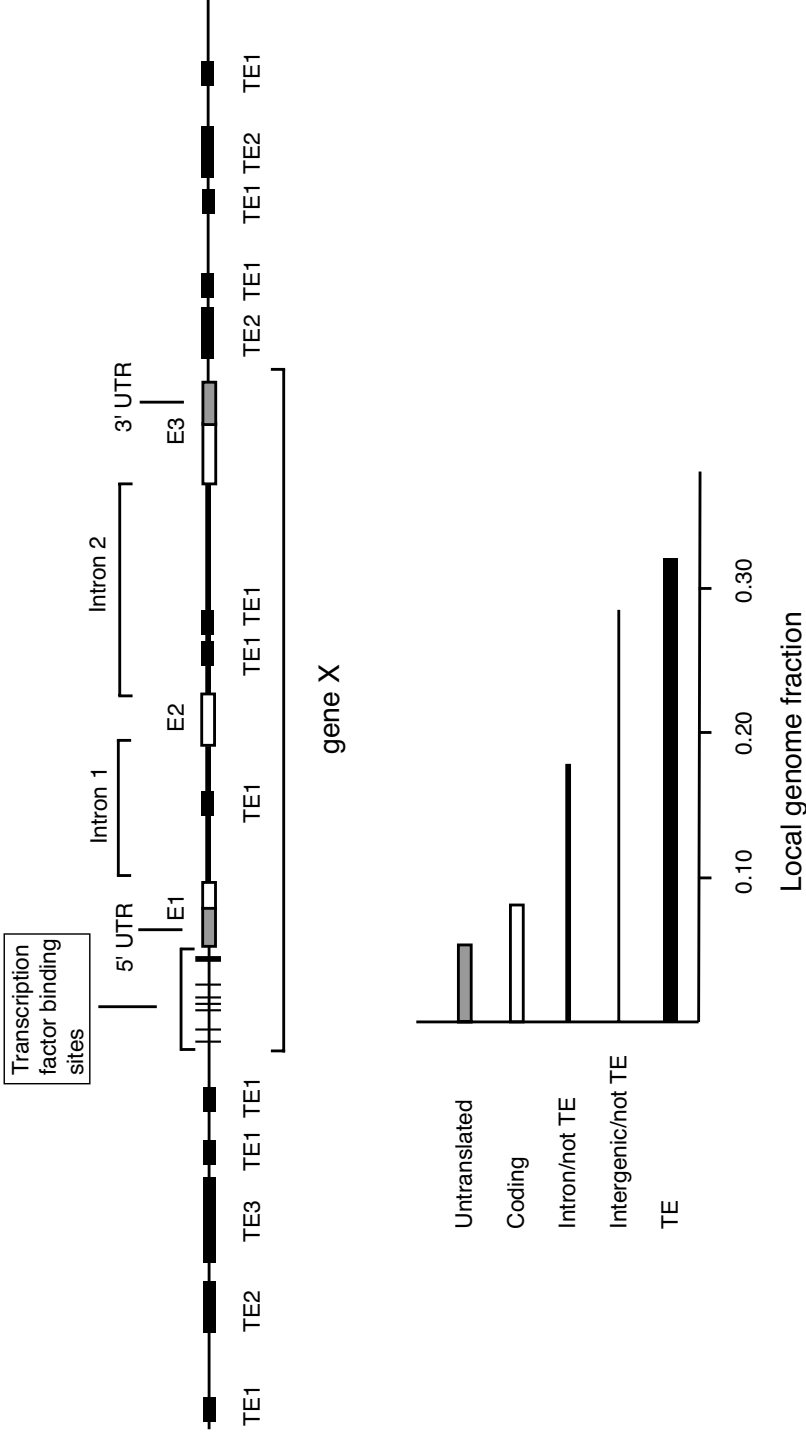


Fig. 14.13. Types of sequences found in eukaryotic genomes. Coding sequences (open boxes within exons E1, E2, and E3) may represent only a small fraction of eukaryotic genomes. Noncoding sequences may be intergenic (thin lines), intronic sequences within genes (thick lines), or untranslated portions of exons (grey boxes). Intersecting vertical lines represent transcription factor binding sites (thin lines) or the core promoter (thick line). Transposable elements of various types (filled boxes) may represent a few percent or the majority of a genomic sequence. The fractional amount of each sequence type in this genome region is shown at the bottom.

Table 14.2. Measures of coding sequence content for eukaryotic model organisms.

Genome	No. Genes	% Coding ^a	Genomic DNA per gene (bp)	Gene size (average bp)	mRNA size ^b (average nt)	Exon size (average bp)	Exons/gene (average)
<i>Saccharomyces cerevisiae</i>	5,500	70	2060	1450	1450	1450	1 ^c
<i>Caenorhabditis elegans</i>	18,400	27	5100	2700	1400	240	6
<i>Drosophila melanogaster</i>	13,600	20	8550	3250	1770	425	4
<i>Arabidopsis thaliana</i>	26,400	26.3	4870	1970	1280	164	5.2
<i>Homo sapiens</i>	25,000	1.2	137,000	27,000	1340	145	9-10

^a % coding computed relative to assembled sequence, which may not include heterochromatin.

^b mRNA size is for the processed transcript, with introns removed.

^c An estimated 240 yeast genes contain one or more introns. 96% of the yeast genes have no introns.

Data: *S. cerevisiae*

(Goffeau et al., 1996)

(C. elegans Sequencing Consortium, 1998)

(Adams et al., 2000)

<http://mips.gsf.de/proj/thal/db/tables/tables-gen-frame.html>

(International Human Genome Sequencing Consortium, 2001; 2004)

Exon/intron statistics for last four organisms above are also available in: Deutsch and Long, 1999.

14.4.1 Gene Prediction from Local Sequence Context

Computational gene prediction in a genomic sequence is an important and complex problem, particularly for eukaryotic genomes. Complete characterization of a gene includes identifying the promoter, all transcription factor binding sites, all introns and exons, the transcriptional start site and termination sites, and sites involved in pre-mRNA processing. These tasks are all computationally nontrivial in their own right, so the complete identification of genes based only upon their local sequence context is rarely achieved for eukaryotes.

Identifying eukaryotic promoter sequences is an important (and difficult) problem. Promoter sequences are recognized by general transcription factors, such as TATA-binding protein, TBP. Not counting regulatory regions, promoters often span a region 200 bp or more upstream of the transcriptional start site at +1. Critical promoter elements are the TATA box, the CAAT box, and the GC box. The TATA box and CAAT box (when present) occur about 25 bp and 100 bp upstream (in the 5' direction) of the transcriptional start site, respectively, but these locations can vary. The GC box may be found about 200 bp upstream of the transcriptional start site. There are “TATA-less” promoters, which may contain multiple copies of the GC box at a variety of positions upstream of the transcriptional start site. Positional weight matrices for the TATA box and the start site are shown in Table 14.3.

A major task for eukaryotic gene prediction is identifying exons and introns from sequence data. Introns contain at least three potential signals recognized by the splicing apparatus for processing mRNA: the 5' donor site, the 3' acceptor site, and the **branch site**. The splicing occurs on an RNA substrate, but gene finding is usually performed on DNA, so we write this as the DNA version. The simplest description of an exon-intron relationship is:



where 5' and 3' refer to the exons immediately flanking a particular intron (not necessarily the exons at the extreme 5' and 3' ends of genes), and the vertical lines separate exons from the intron, which lies between the vertical lines. Conserved dinucleotides GT and AG obviously can occur very frequently by chance, so additional positions are used to define the splice junctions (Table 9.4). These extended sequences provide a better representation (patterns less likely to occur by chance), but as we might expect of consensus methods, these patterns fail to represent completely the range of allowable splice sites. For example, a small proportion of introns contain GC at the 5' splice junction.

Positional weight matrices (expressed as probabilities of occurrence of each base at each position) provide a more complete description for the 5' and 3' splice sites and the branch site (Table 14.3) (Zhang, 1998). Another set of matrices has been compiled for high %G+C genes (Zhang, 1998), and the donor and acceptor sequence preferences for the minority of introns having GC at the 5' splice site have also been determined (Thanaraj and Clark, 2001). With

Table 14.3. Positional weight matrices (profiles) for a variety of features associated with human genes. Positions are labeled by various biological conventions. Entries are probabilities. Intron profiles are for low %G+C loci. All data are taken from Zhang (1998).

Branch profile								5' Splice site profile (low G+C):									
-5	-4	-3	-2	-1	0	+1		-3	-2	-1	0	+1	+2	+3	+4	+5	
A	0.25	0.25	0.00	0.00	0.39	1.00	0.18	A	0.38	0.62	0.12	0.00	0.00	0.71	0.73	0.11	0.21
C	0.19	0.22	0.60	0.02	0.24	0.00	0.33	C	0.31	0.10	0.04	0.00	0.00	0.02	0.06	0.06	0.10
G	0.15	0.17	0.00	0.00	0.32	0.00	0.03	G	0.18	0.12	0.77	1.00	0.00	0.24	0.08	0.75	0.14
T	0.41	0.36	0.40	0.98	0.05	0.00	0.46	T	0.13	0.16	0.07	0.00	1.00	0.03	0.13	0.08	0.55

3' Splice site profile (low G+C):																	
-15	-14	-13	-12	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1	0	+1	
A	0.15	0.14	0.13	0.11	0.10	0.10	0.11	0.12	0.13	0.11	0.10	0.26	0.07	1.00	0.00	0.26	0.24
C	0.24	0.21	0.20	0.22	0.21	0.22	0.25	0.28	0.28	0.25	0.22	0.25	0.55	0.00	0.00	0.11	0.15
G	0.10	0.12	0.10	0.09	0.10	0.09	0.10	0.10	0.08	0.05	0.05	0.15	0.01	0.00	1.00	0.50	0.20
T	0.51	0.53	0.57	0.58	0.59	0.59	0.54	0.50	0.51	0.59	0.63	0.33	0.37	0.00	0.00	0.13	0.41

TATA box profile															
-3	-2	-1	0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+10	+11	
A	0.12	0.06	0.69	0.02	0.70	0.57	0.69	0.42	0.24	0.13	0.20	0.18	0.19	0.18	0.15
C	0.39	0.22	0.04	0.15	0.06	0.06	0.09	0.05	0.18	0.35	0.36	0.31	0.32	0.30	0.30
G	0.35	0.15	0.07	0.11	0.12	0.05	0.16	0.26	0.46	0.41	0.34	0.34	0.31	0.35	0.36
T	0.14	0.57	0.20	0.72	0.12	0.32	0.07	0.27	0.12	0.12	0.11	0.17	0.17	0.18	0.19

Start site profile													
-9	-8	-7	-6	-5	-4	-3	-2	-1	0	+1	+2	+3	
A	0.21	0.15	0.21	0.21	0.18	0.24	0.57	0.31	0.15	1.00	0.00	0.00	0.26
C	0.28	0.38	0.38	0.24	0.35	0.49	0.05	0.43	0.53	0.00	0.00	0.00	0.20
G	0.31	0.27	0.22	0.42	0.27	0.22	0.36	0.17	0.27	0.00	0.00	1.00	0.41
T	0.19	0.20	0.19	0.13	0.20	0.05	0.02	0.10	0.04	0.00	1.00	0.00	0.10

such matrices and the background base frequencies q_a , scoring matrices having $\log_2(p_{ai}/q_a)$ as their elements can be prepared to aid in exon prediction.

For eukaryotic genomic sequences, computational gene prediction tools such as GeneMark, Genie, or Genscan can be employed. Some of these are ab initio gene finders, meaning that they generate a prediction based upon the statistics and signals associated with the input string of the DNA sequence being searched. These methods have associated false-positive and false-negative error rates that affect our ability to identify genes in any organism. Their

accuracy (measured as the average of the sensitivity and specificity of exon prediction) can range from around 45% to 75%, depending upon which tool is employed (Rogic et al., 2001). With metazoan genes, which typically have several exons per gene (Fig. 14.13, Table 14.2), it is clear that the likelihood of correctly predicting an entire gene (all exons) is low (30%–40% correct predictions for HMMGene and Genie applied to a portion of the *Drosophila* genome; Reese et al., 2000).

14.4.2 Exon and Intron Statistics

The sizes of exons and numbers of introns affect gene identification. Average exon and intron statistics for several genomes are presented in Table 14.2. These numbers indicate how complicated genome analysis will be for each organism. For example, only 4% of yeast genes contain introns; therefore, large open reading frames in yeast will be strongly correlated with actual genes, and gene finding will be comparatively easy. In contrast, human protein coding genes contain nine to ten introns on average, and correct gene prediction will require correct identification of about 18 to 20 exon-intron or intron-exon boundaries for each gene. Exon size distribution data can be useful when analyzing genes. Exon size distributions for three different eukaryotes are shown in Fig. 14.14. These distributions are very broad and skewed right (see also Deutsch and Long, 1999). Lengths of candidate exons predicted from 5' and 3' splice signals (see below) can be tested against the exon length distribution for that organism to determine whether the lengths are improbably short or long.

14.4.3 Comparative Methods for Identifying Genes

A simple computational approach for gene finding is a sequence similarity search using alignment applications such as BLASTX to compare translations of genomic sequence in all six reading frames against a protein database. This is particularly appropriate if most genes are not interrupted by introns. If the organism's genes have many introns, the translated string used for searching may not accurately correspond to an exon, which will reduce the sensitivity of the comparison. A similar approach is to compare genomic sequences directly to entries in **EST** (expressed sequence tag) databases. ESTs contain short sequences derived from cDNAs and are frequently determined by single-pass, high-throughput sequencing of thousands of cDNAs derived from particular tissues and particular physiological conditions. ESTs usually contain more sequencing errors than does finished genomic DNA. Also, they frequently only represent the 3' end of the mRNA—not the full-length transcript. Estimates of gene numbers may be confounded by fragments of a single gene that are attributed to separate transcripts and by the failure to recognize pseudogenes. An additional problem for both prokaryotic and eukaryotic genomes is the

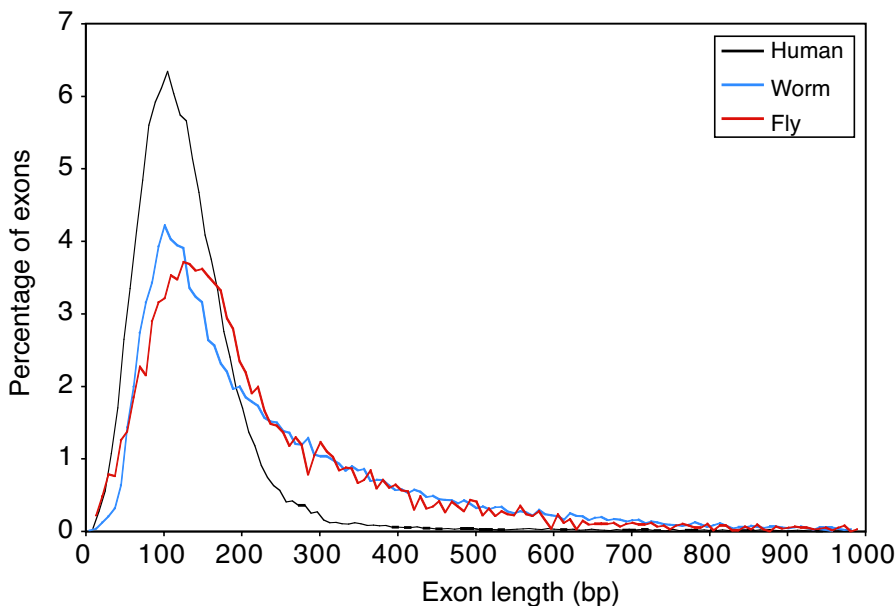


Fig. 14.14. [This figure also appears in the color insert.] Distributions of exon lengths in *H. sapiens*, *C. elegans*, and *D. melanogaster*. Only a small proportion of human exons exceed 300 bp in length, indicating that such a threshold might be used as an additional probabilistic criterion for exon scoring. Reprinted, with permission, from International Human Genome Sequencing Consortium (2001) *Nature* 409:860–921. Copyright 2001 Nature Publishing Group.

absence of significant database matches for substantial fractions of predicted genes (30–40% for early sequencing projects).

Comparative genomics provides an additional approach to gene finding (coding sequences *and* regulatory sequences) that employs several related genomes to reduce the signal-to-noise ratio. After applying gene-finding methods to the genomic sequence of any single organism, we usually do not know whether all exons have been correctly identified, nor do we know what transcription factors regulate the expression of that gene. Now suppose that the corresponding genomic sequences from other related organisms are known. We perform multiple alignment on their corresponding regions. Since the gene and its regulatory sequences have been evolutionarily selected for function, we expect to find a higher level of sequence conservation within the coding and regulatory sequences than in the introns or “spacer” regions between transcription factor binding sites.

This concept is diagrammed in Fig. 14.15. After multiple sequence alignment, correct translational start sites, stop sites, and exon boundaries can be inferred from the “consensus” assignment for the whole set. Regulatory sequences are particularly difficult to identify from a single genome because

the motifs may be short and may display considerable sequence degeneracy at positions within the sequence pattern. If we were to use PWMs to search for regulatory sequences (Chapter 9), there would typically be a substantial false positive error frequency. But with aligned genomic sequences, the regulatory sequences become more apparent because of their conserved sequences and positions relative to the coding sequences. Identifying regulatory sequences by comparing sequences from corresponding genomic regions of related organisms is called **phylogenetic footprinting** (Gumucio et al., 1992; see Zhang and Gerstein, 2003, for a review).

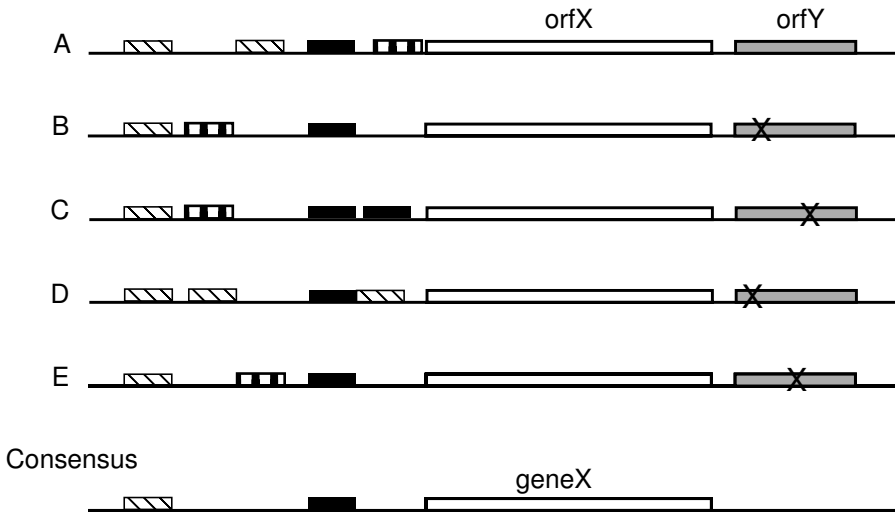


Fig. 14.15. Phylogenetic footprinting using related organisms A, B, . . . , E. Open boxes represent coding sequences, grey boxes represent untranslated ORFs, and black or striped boxes represent candidate regulatory elements in orthologous gene regions. “X” indicates a stop codon. In any one genome, it may be hard to determine which regulatory elements are functional, but conserved elements shared by the whole set most likely represent the actual functional elements. Similar logic can be used to identify functional ORFs. Although orfY appears in A, its absence in the other strains (note stop codons) suggests that it may not be functional.

The power of such methods is illustrated by the comparative study of *S. cerevisiae* and three close relatives (Cliften et al., 2003; Kellis et al., 2003). As we mentioned earlier, yeast gene identification is relatively easy, because of the 6062 ORFs listed in the *Saccharomyces* Genome Database, only 240 were indicated to have one or more introns. Even with this simple genome, the comparative genomic approach showed that about 500 of the approximately 6000 previously annotated “genes” were probably not genes at all. The phylogenetic footprinting approach identified 40 new regulatory motifs in addition to many of the 55 regulatory motifs that had previously been identified (Kellis

et al., 2003). With the availability of additional genome sequences throughout the phylogenetic tree, this comparative approach will be a powerful addition to other methods of computational gene analysis.

14.4.4 Gene Numbers

The size, number of genes, and fraction transcribed are important properties distinguishing genomes. Genome sizes and estimates of numbers of genes already give us significant information (Table 14.1). We start by noting that there is no clear relationship between the amount of DNA and the genetic complexity of an organism. For example, *Arabidopsis* has more genes than *Drosophila* but less DNA. Next, note the number of genes contained in the *Mycoplasma genitalium* genome. This number (about 500) gives us an upper bound for the minimum number of genes required to operate an autonomous, free-living organism. Experimental gene knockout studies on this organism suggest that the minimal genome for autonomous existence is in the range of 250–350 genes (Fraser et al., 1995). If we compare the numbers of genes of *M. genitalium* and *E. coli*, we see that *E. coli* has about 8 times the number of genes found in *M. genitalium*. This may mean that *E. coli* has accumulated genes that are useful for growth under a broader range of environmental conditions. Genes for the utilization of particular sugars such as lactose and xylose exemplify this phenomenon (genes in the lactose operon are not significantly expressed in the absence of lactose). If we compare the numbers of genes of *S. cerevisiae* (a eukaryote) and *E. coli* (a prokaryote), we find numbers that are similar, suggesting that the eukaryotic lifestyle is not just a matter of having more genes but rather having particular sets of gene types. For example, there are 1105 yeast genes that are essential for growth in rich medium (Giaever et al., 2002) compared with an estimated minimal genome of 250–350 for *Mycoplasma*. Of course, numbers of genes are insufficient to represent the complex organization of eukaryotic regulatory networks.

Comparison between metazoan genomes is instructive. For example, *C. elegans* (a nematode “worm”) is a very simple organism that as an adult hermaphrodite consists of 959 somatic cells. (This is the actual *cell number*, not the number of cell *types*.) Its genome is predicted to contain about 18,400 genes. The adult *D. melanogaster* is a much more complex organism, both morphologically and behaviorally, yet it is predicted to have 13,600 genes. As indicated above, we conclude that it is not just the number of genes that determines the complexity of an organism but rather the types of genes or the complexity of the interactions between genes and gene products that matters.

A comparison of the human genome with that of *Arabidopsis thaliana* is also informative. Both are multicellular eukaryotes, and both contain roughly the same number of genes. We tend to view humans from the standpoint of behavioral complexity, while we say that plants just “vegetate.” However, plants have additional biochemical complexity that humans do not possess (e.g., human cells cannot manufacture all of the essential amino acids, nor can they

engage in photosynthesis), and thus the similarity in number of genes is perhaps understandable. Note, however, that the human genome is about 25 times larger than the *Arabidopsis* genome, even though the numbers of genes are comparable. This size difference could be explained by a number of hypotheses, including the suggestion that more of the human genome is devoted to gene control, or alternatively the more trivial explanation that nonfunctional DNA has accumulated to a greater extent in the human genome. We saw earlier (Section 14.2) that the human genome contains a higher proportion of transposable elements than does the genome of *Arabidopsis*.

14.5 Predicted Proteome

We now consider how to annotate and catalog predicted genes. Our goal is to assign one or more functions to each predicted polypeptide. Having annotated genes to the extent possible, the next job is to catalog them and to make a list of all different types of genes by functional category. This indicates the repertoire of functional pathways available to the organism.

14.5.1 Assigning Gene Function by Orthology

We must specify what is meant by “function.” Recognizing a need for a common descriptive vocabulary and classification scheme for the various concepts subsumed by the term “function,” the Gene Ontology Consortium has defined three families of hierarchical descriptors corresponding to molecular functions, biological processes, and cellular components (Gene Ontology Consortium, 2004). In terms of *molecular function*, a protein might, for example, be a permease, a phosphatase, or a DNA-dependent ATPase. Although they are precise designations, these descriptions do not describe the *processes* in which the proteins participate. For example, the permease might be involved in utilization of a sugar, the phosphatase might be part of a signaling pathway, and the DNA-dependent ATPase might be a helicase involved in DNA replication. In terms of *cellular components* to which the proteins belong, the permease might be associated with the plasma membrane, and the helicase (DNA-dependent ATPase) might be associated with the replication fork. Clearly, a complete description of the function of any gene product will include (but not necessarily be limited to) a specification of its molecular function, biological process, and cellular location.

Ordinarily, protein functions cannot be deduced from first principles using the predicted polypeptide sequences. Instead, functions are predicted based upon patterns of gene expression from microarray data (see Chapter 11) or by sequence similarity to other proteins whose functions have been characterized biochemically. Assume that we are starting with the collection of predicted polypeptide sequences for a particular genome. Each one of these polypeptides can be used as a query sequence to search appropriate databases. The hits

(database matches) that display significant similarity with the query sequence are retrieved and listed. If the similarity is sufficiently high and if the functions of some of the hits are known, then it may be reasonable to attribute some of their functional annotations to the query sequence.

As we indicated in our discussion of alignment (Chapter 6), sequence similarity means the degree of agreement of the character states at positions of one sequence aligned with another. (Remember that similarity over short segments of two strings need not imply an evolutionary relationship between two proteins: the agreement could have occurred by chance.) We indicated in Section 6.1 that genes or proteins that have descended from a common ancestor are said to be **homologs**, which are expected to show sequence similarity, provided that the evolutionary distance separating them is not too great. Recall that if gene homologs g_B and g_C (descended from gene g_A) are observed in organisms B and C, respectively, then g_B and g_C are said to be **orthologs**. The functions of orthologs are often the same as the function of the ancestral gene g_A , so for annotation of sequences, orthologous relationships are particularly helpful (i.e., if the function of g_B is known but that of g_C is not, the function of g_C is likely to be the same if g_B and g_C are orthologs). But suppose that after the lineage of organism B split from the C lineage, duplications of g_B occurred, leading to production of g_{B1} and g_{B2} . Genes g_{B1} and g_{B2} , products of intragenomic duplication, are said to be **paralogs**. One of these (g_{B1} , for example) may retain the same function as g_C , but g_{B2} may have acquired a different function as a result of mutation and selection. This is possible because the other copy, g_{B1} , is still available to supply the original function. Gene g_C (function unknown) might show sequence similarity to both paralogs g_{B1} and g_{B2} . In this case, the match with g_{B1} would provide a correct functional assignment, but the match with g_{B2} would not.

Identifying orthologs is an important activity in comparative genomics because experimentally determined functions of genes in tractable model organisms can be associated with their orthologs found in newly examined organisms. For example, ribosomes in *E. coli* have been shown experimentally to contain 52 different protein species. Identification and characterization of these proteins required many years of genetic and biochemical research. Because ribosomal functions are highly conserved, we expect orthologs of most of these 52 proteins to appear in most bacteria. Orthologous relationships make it unnecessary to repeat all of the extensive “wet lab” work to identify and characterize ribosomal proteins for each new eubacterium whose genome is sequenced.

As increasing numbers of complete genome sequences become available, more and more orthologous relationships can be established. A consistent way of defining and cataloging these relationships is through the use of **clusters of orthologous genes** (or **COGs**; Tatusov et al., 1997). Originally defined for prokaryotes (for which many more complete genome sequences are available), they have been extended to include orthologous groups within eu-

karyotic genomes as well. These groups are called KOGs. COGs are produced as follows:

1. Start with a data set containing predicted proteins encoded in a set of complete genomes (43 prokaryotic genomes in a recent release).
2. Perform an “all-against-all” series of BLASTP comparisons using the predicted protein sequences. This means that each protein sequence in an organism is used as a query sequence for a BLASTP comparison against all other protein sequences in all of the organisms (including the one from which it came).
3. Multiple “hits” from within-organism comparisons are indicative of paralogs, which are treated as a single group—not as multiple instances.
4. Identify best hits for each gene in each organism with comparable genes in other organisms.
5. Define a COG as the set of mutually consistent best hits found between genomes in a minimum of three organisms. Genes correspond to nodes, and the mutual relationships define the edges of a triangular graph defining the minimal COG.
6. Expand the graph of any COG by merging graphs with COGs that share edges in common with it.

This process is illustrated in Fig. 14.16. Because genome D may be evolutionarily remote from A, the similarity between a particular protein in D and its ortholog in A may not be recognized, whereas its similarity to the corresponding genes in B and C is significant. The last step allows orthology between the genes in D and A to be identified, because of their relationships to orthologs in B and C.

Each COG corresponds to a particular function represented in a collection of genomes. There are currently 3300 COGs. Some functions may appear in all genomes, whereas others have a more restricted distribution. For example, small ribosomal subunit protein S3 appears in all 43 reference genomes, but ribonuclease I appears in only three. COGs have been organized into the functional categories listed in Table 14.4.

14.5.2 Assigning Gene Function by Patterns of Occurrence

In the previous section, we described the assignment of a gene’s function based upon *orthology*, which was recognized on the basis of sequence similarity between gene products from two different organisms. This type of functional assignment includes all three aspects of function—molecular activity, biochemical process, and cellular localization—provided those annotations are available for one member of the orthologous pair. However, there are proteins that participate in the same functional process but are not orthologs. (For example, DNA polymerases and helicases both may be involved in DNA replication). How can we recognize function at the level of process or cellular location if the participating proteins are not orthologs or if there are no

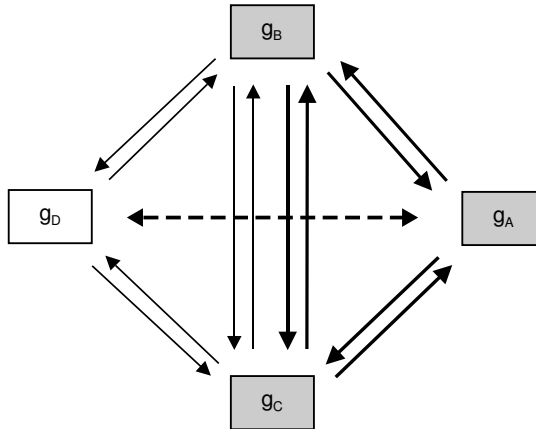


Fig. 14.16. Definition of COGs. Orthologs of gene g in organisms A, B, and C can be recognized because the predicted polypeptides are each other's best "hit" for every genome pair (indicated by arrows). The minimal COG consists of this type of pairwise relationship for a particular gene represented by orthologs in three genomes (shaded boxes). Another COG including genomes B, C, and D might also be recognized. Even though g_D and g_A may have diverged from each other enough to make their orthologies hard to detect (dashed line), the common side shared by these two COGs is sufficient to join them, thus placing these orthologs from all four organisms into relationship.

suitable annotated orthologs available? As we mentioned earlier, this situation is not uncommon: 30% or more of the genes found in a newly sequenced organism may have *no* database matches.

At least three genomic methods for identifying shared processes are particularly successful for prokaryotes and also are applicable to eukaryotes. These methods rely on patterns of gene distribution in three different contexts: distribution of genes within genomes (**gene neighbors**), association of independent polypeptides in one organism with **gene fusions** in another, and distribution of genes among a reference collection of organisms (**phylogenetic profiles**). These methods can be applied individually, together, or in combination with experimental approaches, such as expression profiling, to identify *networks* of genes (not necessarily orthologs) that participate in a shared functional process.

The *gene neighbors* approach relies on the observation that genes having related functions tend to be more closely linked genetically than genes having unrelated functions. This association in map positions is particularly strong for prokaryotes, whose genes are commonly organized into operons. (Even the eukaryote *C. elegans* has 25% of its genes within operons.) The close proximity of related genes allows the sharing of promoter elements for coordinated gene expression. Suppose that in genome i we observe that gene A (for which the functional process is known) is located next to gene B (for which the func-

Table 14.4. Summary of clusters of orthologous genes (COGs) by functional categories (Tatusov et al., 2000). Data from <http://www.ncbi.nlm.nih.gov/COG/>.

Code	Number of COGs	Number of Domains	Description
<i>Information storage and processing</i>			
J	217	6449	Translation, ribosomal structure, and biogenesis
K	132	5438	Transcription
L	184	5337	DNA replication, recombination, and repair
<i>Cellular processes</i>			
D	32	842	Cell division and chromosome partitioning
O	110	3165	Post-translational modification, protein turnover, chaperones
M	155	4079	Cell envelope biogenesis, outer membrane
N	133	3110	Cell motility and secretion
P	160	5112	Inorganic ion transport and metabolism
T	97	3627	Signal transduction mechanisms
<i>Metabolism</i>			
C	224	5594	Energy production and conversion
G	171	5262	Carbohydrate transport and metabolism
E	233	8383	Amino acid transport and metabolism
F	85	2364	Nucleotide transport and metabolism
H	154	4057	Coenzyme metabolism
I	75	2609	Lipid metabolism
Q	62	2754	Secondary metabolite biosynthesis, transport, and catabolism
<i>Poorly characterized</i>			
R	449	11948	General function prediction only
S	750	6416	Function unknown

tional process is unknown). Moreover, suppose that these genes also appear in close proximity in genomes j, k, l, \dots . The conserved neighboring relationships across a range of organisms suggests that these two genes may share in the same functional process (Dandekar et al., 1998; Overbeek et al., 1999).

The second method depends upon the observation that proteins often contain several functional domains that fold in a structurally independent manner. For example, genes *trpG* and *trpD*, which are involved in the biosynthesis of tryptophan, encode glutamidotransferase and anthranilate phosphoribosyl transferase activities, respectively. The products of these two genes appear on separate polypeptide chains in *Serratia marcescens*. However, these activities appear together on a single polypeptide chain (a product of the *trpD* gene) in *E. coli*. Similarly, in *E. coli*, the DNA polymerase III holoenzyme contains ten

subunits. One of these is the α subunit, which is encoded by *dnaE* and displays polymerase activity, and another is the ϵ subunit, encoded by *dnaQ* and having 3'→5' exonuclease activity. In *Bacillus subtilis*, the replication polymerase PolIII α contains the domains for polymerase and 3'→5' exonuclease fused on a single polypeptide chain. Notice that in both of these examples, each pair of polypeptides (or domains) functions in the same process (a result of experimental observation). Fusion of these domains into a single polypeptide chain is predicted to facilitate the process on biochemical grounds. If we reverse the logic, we can predict that *if two polypeptides A_i and B_i occur separately in organism i but each is similar to different portions of a larger polypeptide in organism j (i.e., gene g_j appears to be a fusion of regions encoding polypeptides resembling A_i and B_i), then A_i and B_i probably participate in the same biological process* (Enright et al., 1999). Note that genes A_i and B_i are not similar to each other, so their functional assignment does not depend upon orthology. In addition, appearance of these domains in a single polypeptide chain, where a degree of interaction will occur, suggests that these domains also physically interact when they occur on separate polypeptide chains. Tests of this method using independently annotated genes showed that 32%–68% of genes associated by domain fusion shared keywords in their annotations (for yeast and *E. coli*, respectively), while pairs of genes selected at random shared keywords in 14%–15% of their annotations (Marcotte et al., 1999).

The third method, identification of phylogenetic profiles (Pellegrini et al., 1999), employs statistical tools similar to those that we used for classification and clustering. The procedure is, in a sense, the “transpose” of what we were doing before. Before, we used matrices that looked like the one below to summarize the properties of OTUs :

OTU	characters				
	i	ii	iii	iv	... n
1	0	1	1	0	... 0
2	1	0	1	1	... 0
3	0	1	1	0	... 0
4	0	1	1	0	... 1
.
.
m	1	0	0	0	... 0

Instead of using anatomical characters, we could use the presence (1) or absence (0) of particular proteins p_1, p_2, \dots, p_n in m completely sequenced genomes (the genomes *must* be completely sequenced) as our characters. At least some of the p_i should be annotated functionally in some model organism.

OTU	p_1	p_2	p_3	p_4	\dots	p_n
1	0	1	1	0	\dots	0
2	1	0	1	1	\dots	0
3	0	1	1	0	\dots	0
4	0	1	1	0	\dots	1
⋮	⋮	⋮	⋮	⋮	\dots	⋮
⋮	⋮	⋮	⋮	⋮	\dots	⋮
m	1	0	1	1	\dots	0

Now transpose the matrix to represent the distribution of each protein across taxa. Each row represents the phylogenetic profile of a particular protein. Remember, some of these proteins p_i will have been functionally annotated in one or more genomes.

Proteins	OTU ₁	OTU ₂	OTU ₃	OTU ₄	\dots	OTU _{m}
p_1	0	1	0	0	\dots	1
p_2	1	0	1	1	\dots	0
p_3	1	1	1	1	\dots	1
p_4	0	1	0	0	\dots	1
⋮	⋮	⋮	⋮	⋮	\dots	⋮
⋮	⋮	⋮	⋮	⋮	\dots	⋮
p_n	0	0	0	1	\dots	0

Boldface entries highlight the vectors of two proteins having similar profiles.

The key idea for using such data is that genes encoding proteins participating in the same functional pathway tend to be coordinately present or absent in a collection of genomes. The phylogenetic profile of a protein is the n -dimensional vector representing its presence (1) or absence (0) in each of n genomes (OTUs). Proteins having similar profiles are inferred to be functionally linked. In the illustration above, proteins p_1 and p_4 are expected to be functionally related, and if the functional process is known from annotation of one of them, that process will be attributed to the other. Or, if a new protein p_x with no database matches has the profile

$$p_x \left| \begin{array}{cccccc} \text{OTU}_1 & \text{OTU}_2 & \text{OTU}_3 & \text{OTU}_4 & \dots & \text{OTU}_m \\ 0 & 1 & 0 & 0 & \dots & 1 \end{array} \right.$$

then it would be inferred to be functionally related to p_1 and p_4 . This approach has been evaluated for vectors generated from 16 genomes by examining the number of pairs of proteins matched by their phylogenetic profiles that also share keywords from prior independent annotations. This number was compared with the number of keyword matches for gene pairs selected at random and found to be substantially higher (Pellegrini et al., 1999).

14.5.3 Gene Content Within and Between Organisms

Our attempts to conceptualize the complex biological world raise questions such as, “What is an archaeobacterium?”, “What is a eukaryote?”, or “What

is an animal?” Traditional systematics has employed large numbers of phenotypic characters to answer these questions, guided by an evolutionary perspective. With the availability of whole-genome sequences, it is informative to ask these questions again from the perspective of gene content. After all, the genes and the networks of interactions between genes and gene products give rise to those phenotypes that we associate with each organism. Within-genome analysis gives a picture of what biochemical or physiological functions the cells of a particular organism can perform. Between-genome comparisons reveal gene categories that are or are not shared by taxa (e.g., What gene categories are shared by vertebrates? What gene categories are found in animals but not fungi?).

Comparison of gene contents among organisms requires a unified conceptual framework for categorizing genes. For example, genes might be grouped into the functional categories defined by COGs (Table 14.4). Some of the higher-level categories for each of the three conceptual definitions of function proposed by the Gene Ontology Consortium (see Section 4.5.1) are shown in Table 14.5. To see how these definitions apply, consider just three categories of genes from *H. sapiens* and *Arabidopsis thaliana* (IHGSC, 2001) using molecular function to define categories. In the “defense and immunity” class, there are over a thousand human genes, while *Arabidopsis* has only about 200 such genes. In the “intracellular signaling” category, humans and *Arabidopsis* have similar numbers of genes (about 1800 and 1700, respectively). For “general metabolism,” *Arabidopsis* has over 1000 more genes than humans (about 4600 for *Arabidopsis* compared with 3300 for humans). The comparisons for the first and last functional categories are as expected from the known biology of plants and animals (e.g., plants don’t produce antibodies, and humans don’t photosynthesize or produce cellulose).

An alternative way of comparing proteome content across organisms is by the inventory of domains found among the proteins encoded by their genomes. Many proteins can be described in terms of their constituent motifs, profiles, and domains. A protein **motif** is a short polypeptide sequence pattern associated with a particular function. An example is the ATP/GTP-binding site motif A (amino acid sequence (A/G)-x-x-x-x-G-K-(S/T), where x is any amino acid residue). A domain is a coherent structural polypeptide unit that is associated with a particular function. A domain may include one or more motifs. A **profile** is a probabilistic description of a protein sequence pattern, perhaps in terms of a positional weight matrix. Table 14.6 provides a comparison of the domain content of humans, pufferfish, *C. elegans*, and fruitflies, listing only some of the more common domains. The numbers of different types of domains encoded provide a measure of the genetic capabilities of the organisms. For example, note the lower number of homeobox domains in *C. elegans* compared with the vertebrates.

Identifying genes shared by animals but absent in other eukaryotes is a partial answer to the question, “What is an animal?” Sets of proteins shared by two or more organisms are determined by using alignment scores obtained

Table 14.5. Examples of gene ontology high-level categories (Gene Ontology Consortium, 2004). Information taken from http://ftp.geneontology.org/pub/go/GO_slims/goslim_generic.go.

Biological process	Cellular component	Molecular function
Behavior	Chromosome	Antioxidant activity
Cell communication	Cilium	Apoptosis regulator activity
Cell recognition	Cytoplasm	Binding
Cell-cell signaling	Cytoplasmic chromosome	Cell adhesion molecule activity
Host-pathogen interaction	Cytoplasmic vesicle	Chaperone activity
Endogenous stimulus response	Cytoskeleton	Chaperone regulator activity
External stimulus response	Cytosol	Defense/immunity protein activity
Signal transduction	Endoplasmic reticulum	Catalytic activity
Cell growth/maintenance	Endosome	Enzyme regulator activity
Cell cycle	Golgi apparatus	Motor activity
Cell growth	Lipid particle	Protein stabilization activity
Cell organization/biogenesis	Microtubule organizing center	Protein tagging activity
Cell proliferation	Mitochondrion	Signal transducer activity
Chem-mechanical coupling	Peroxisome	Nutrient reservoir activity
Cell homeostasis	Plastid	Structural molecule activity
Metabolism	Ribosome	Transcription regulator activity
Response to stress	Vacuole	Translation regulator
Transport	Nucleus	Transporter activity
Death	Nuclear chromosome	Triplet codon/amino acid adapter
Development	Nuclear membrane	
Cell differentiation	Nucleolus	
Embryonic development	Nucleoplasm	
Morphogenesis	Plasma membrane	
	Thylakoid	

Table 14.6. Comparison of animal proteome complexities based upon selected domain content. Domains are listed in decreasing order of frequency among human proteins. Values correspond to the approximate number of instances of each domain in each organism.^a

InterPro Accession	Name	Number of each domain type in:		
		<i>H. sapiens</i>	<i>F. rubripes</i>	<i>C. elegans D. melanogaster</i>
000822	Zn-finger, C2H2 type	800	500	200
000276	Rhodopsin GPCR	750	500	400
003006	Immunoglobulin/MHC	750	500	50
000719	Eukaryotic protein kinase	550	650	400
003593	ATPase	400	350	250
001909	KRAB box	300	0	0
001680	G-protein beta WD-40 repeat	300	300	150
001841	Zn-finger, RING	300	300	150
001849	Plekstrin-like domain	250	300	50
000504	RNA-binding RNP-1	250	200	100
002110	Ankyrin	250	250	100
001356	Homeobox	250	300	100
001611	Leucine-rich repeat	250	250	50
000561	EGF-like domain	250	250	50
002048	Calcium-binding EF-hand	250	250	100
001452	SH3 domain	200	250	50

^a Gene numbers for *H. sapiens*, *F. rubripes*, *C. elegans*, and *D. melanogaster* were taken to be 27,000, 30,000, 18,400, and 13,600, respectively. Gene numbers for each domain type were calculated based upon percentages listed in Mouse Genome Sequencing Consortium (2002). Gene numbers were rounded off to the nearest 50, to reflect current uncertainty in gene numbers.

by using FASTA or some species of BLAST. This type of comparison can be performed in a pairwise manner (e.g., what fraction of *Drosophila* genes are also found in *C. elegans*, and vice versa). There are $n(n-1)/2$ possible pairwise comparisons between n organisms, and for each of these, there are two choices for the reference genome. For example, in a *Drosophila/C.elegans* comparison, we could state that 7.8% of the *Drosophila* genes are found in *C. elegans* or that 5.9% of *C. elegans* genes are found in *Drosophila*.

The number of shared proteins depends upon two factors, one statistical and one conceptual. The conceptual issue is whether comparisons are to be made at the level of proteins or from their functional domains. If comparisons of complete genes are made, then the result may be confounded by gene fusions. (We have noted this phenomenon in the previous section.) The statistical issue is the cutoff score chosen as the criterion for orthology. The score might be an E-value from an all-versus-all BLAST search. Are two genes orthologs when $E < 10^{-100}$? When $E < 10^{-50}$? When $E < 10^{-10}$? Obviously, with $E < 10^{-10}$, more orthologs will be identified than if $E < 10^{-100}$ were chosen.

For example, the number of *Drosophila* genes shared by the fly, worm, and yeast was determined (Rubin et al., 2000). This is one approximate measure of the gene set shared by eukaryotes. At a cutoff of $E < 10^{-100}$, there are 435 fly genes shared with the worm and yeast. Pairwise comparison of yeast with the worm and the worm with yeast yielded 330 and 370 shared genes, respectively, at a similar cutoff. These results would suggest that the “core proteome” distinguishing eukaryotes is approximately 300–400 genes. When shared genes at a cutoff of $E < 10^{-50}$ (about 1000 genes) were divided into functional categories, the number of genes in each category was found to be about the same in both the worm and yeast (e.g., approximately equal numbers of genes involved in nucleic acid metabolism in both organisms and approximately equal numbers of genes involved in signal transduction in both organisms) (Chervitz et al., 1998). This would be expected for those core functions shared by eukaryotic cells.

Another between-genome study compared the human proteome to proteomes of the fly, worm, and yeast (IHGSC, 2001). Reciprocal best hits instead of numerical cutoff scores were used to define orthologous groups of genes that were shared among all four organisms. (These groups included paralogs.) There were 1308 such groups. When orthologs only were considered (one gene in each of the four organisms), this number was reduced to 564 proteins, which would correspond to the eukaryotic “core proteome” defined by this clustering criterion (which is different from the cutoff criterion described in the previous paragraph). This number (564) is nevertheless of the same order of magnitude as the size of the core proteome estimated from genes shared by the fly, worm, and yeast. Over half of the proteins in this group (where functional assignments could be made) were associated with “housekeeping” functions (e.g., transcription, translation, DNA replication, etc.).

Rather than searching for genes shared by a group of organisms, we can also look for genes that distinguish one group from another. For example, genes shared among archaeobacteria but not with eubacteria or eukaryotes have been sought (Graham et al., 2000). In this study, an all-versus-all comparison was made among genes in *Methanococcus jannaschii*, *Methanobacterium thermoautotrophicum*, *Archaeoblobus fulgidus*, and *Pyrococcus* species. Genes having $E < 10^{-20}$ were clustered (including orthologs and paralogs). Graham et al. (2000) identified 351 clusters containing no matches to eubacterial or eukaryotic genes. These represented 1149 protein-coding genes. Of these, 81% were hypothetical proteins (i.e., having no functional annotation). Of the 351 clusters, 20% were represented in each of the four genomes. The archaeal-specific genes constitute a genome signature for this domain of life, and when functions become known, these genes may help define what an archaeobacterium is at the biochemical level.

14.6 New Biological Perspectives from Genomics

Genome sequences and massively parallel gene expression data added to prior biochemical and genetic knowledge create a more comprehensive way of thinking about the biology of organisms. Now it is possible to examine organisms at a number of different levels of generalization, ranging from the very particular (Does the organism encode anthranilate synthase?) to the very general (How are human developmental functions regulated and expressed?). Given the breadth of genome sequencing efforts still in progress, it is premature to designate the present time as the “Post-Genome Era,” but it is clear that questions concerning proteomics and interaction networks are becoming more prominent. This means that genes and gene products soon will be understood from an integrated perspective. Any student of genomics (undergraduate, graduate, post-doctoral, or professor) needs to develop skills to think at all levels of generalization and to be equipped with the computational skills to do so. The reward for learning these skills is the ability to think and work in one of the most revolutionary scientific areas of our time. The result will be an appreciation and understanding of how biology “works” to produce living organisms from an inorganic world.

References

- Adams MD et al. (2000) The genome sequence of *Drosophila melanogaster*. *Science* 287:2185–2195.
- The Arabidopsis Genome Initiative (2000) Analysis of the genome sequence of the flowering plant *Arabidopsis thaliana*. *Nature* 408:796–815.
- Bailey JA, Gu Z, Clark RA, Reinert K, Samonte RV, Schwartz S, Adams MD, Myers EW, Li PW, Eichler EE (2002) Recent segmental duplications in the human genome. *Science* 297:1003–1007.

- C. elegans* Sequencing Consortium (1998) Genome sequence of the nematode *C. elegans*: A platform for investigating biology. *Science* 282:2012–2021.
- Chervitz SA, Aravind L, Sherlock G, Ball CA, Koonin EV, Dwight SS, Harris MA, Dolin-ski K, Mohr S, Smith T, Weng S, Cherry JM, Botstein D (1998) Comparison of the complete protein sets of worm and yeast: Orthology and divergence. *Science* 282:2022–2028.
- Cliften P, Sudarsanam P, Desikan A, Fulton L, Fulton B, Majors J, Waterston R, Cohen BA, Johnson M (2003) Finding functional features in *Saccharomyces* genomes by phylogenetic footprinting. *Science* 301:71–76.
- Dandekar T, Snel B, Huynen M, Bork P (1998) Conservation of gene order: A fingerprint of proteins that physically interact. *Trends in Biochemical Sciences* 23: 324–328.
- Deutsch M, Long M (1999) Intron-exon structures of eukaryotic model organisms. *Nucleic Acids Research* 27:3219–3228.
- Dietrich FS, Voegeli S, Brachat S, Lerch A, Gates K, Steiner S, Mohr C, Pohlmann R, Luedi P, Choi S, Wing RA, Flavier A, Gaffney TD, Philippsen P (2004) The *Ashbya gossypii* genome as a tool for mapping the ancient *Saccharomyces cerevisiae* genome. *Science* 304:304–307.
- Durand D, Sankoff D (2003) Tests for gene clustering. *Journal of Computational Biology* 10:453–482.
- Enright AJ, Illopoulos I, Kyrpides NC, Ouzounis CA (1999) Protein interaction maps for complete genomes based on gene fusion events. *Nature* 402:86–90.
- Fraser CM, Gocayne JD, White O, Adams MD, Clayton RA, Fleischmann RD, Bult CJ, Kerlavage AR, Sutton G, Kelley JM, Fritchman JL, Weidman JF, Small KV, Sandusky M, Fuhrmann J, Nguyen D, Utterback TR, Saudek DM, Phillips CA, Merrick JM, Tomb J-F, Dougherty BA, Bott KF, Hu P-J, Lucier TS, Peterson SN, Smith HO, Hutchison CA III, Venter JC (1995) The minimal gene complement of *Mycoplasma*. *Science* 270:397–403.
- Fraser CM, Casjens S, Huang W-M, Sutton GG, Clayton R, Lathigra R, White O, Ketchum KA, Dodson R, Hickey EK, Gwinn M, Dougherty B, Tomb J-F, Fleischmann RD, Richardson D, Peterson J, Kerlavage AR, Quackenbush J, Salzberg S, Hanson M, van Vugt R, Palmer N, Adams MD, Gocayne J, Weidman J, Utterback T, Watthey L, McDonald L, Artiach P, Bowman C, Garland S, Fujii C, Cotton MD, Horst K, Roberts K, Hatch B, Smith HO, Venter JC (1997) Genomic sequence of a Lyme disease spirochaete, *Borrelia burgdorferi*. *Nature* 390:580–586.
- Gene Ontology Consortium (2004) The Gene Ontology (GO) database and informatics resource. *Nucleic Acids Research* 32:D258–D261.
- Giaever G et al. (2002) Functional profiling of the *Saccharomyces cerevisiae* genome. *Nature* 418:387–391.
- Goffeau A, Barrell BG, Bussey H, Davis RW, Dujon B, Galibert F, Hoheisel JD, Jacq C, Johnston M, Louis EJ, Mewes HW, Murakamai Y, Philippsen P, Tettelin H, Oliver SG. (1996) Life with 6000 genes. *Science* 274:546–567.

- Graham DE, Overbeek R, Olsen GJ, and Woese CR (2000) An archaeal genomic signature. *Proceedings of the National Academy of Sciences USA* 97:3304–3308.
- Gregory SG et al. (2002) A physical map of the mouse genome. *Nature* 418:743–750.
- Gumucio DL, Heilstedt-Williamson H, Gray TA, Tarle SA, Shelton DA, Tagle DA, Slightom JL, Goodman M, Collins FS (1992) Phylogenetic footprinting reveals a nuclear protein which binds to silencer sequences in the human gamma and epsilon globin genes. *Molecular and Cell Biology* 12:4919–4929.
- International Human Genome Sequencing Consortium (IHGSC) (2001) Initial sequencing and analysis of the human genome. *Nature* 409:860–921.
- International Human Genome Sequencing Consortium (IHGSC) (2004) Finishing the euchromatic sequence of the human genome. *Nature* 431:931–945.
- Karlin S, Campbell AM, Mrázek J (1998) Comparative DNA analysis across diverse genomes. *Annual Review of Genetics* 32:185–225.
- Kellis M, Birren BW, Lander ES (2004) Proof and evolutionary analysis of ancient genome duplication in the yeast *Saccharomyces cerevisiae*. *Nature* 428:617–624.
- Kellis M, Patterson N, Endrizzi M, Birren B, Lander ES (2003) Sequencing and comparison of yeast species to identify genes and regulatory elements. *Nature* 423:241–254.
- McLysaght A, Hokamp K, Wolfe KH (2002) Extensive genomic duplication during early chordate evolution. *Nature Genetics* 31:200–204.
- Marcotte EM, Pellegrini M, Ng H-L, Rice DW, Yeates TO, Eisenberg D (1999) Detecting protein function and protein-protein interactions from genome sequences. *Science* 285:751–753.
- Mouse Genome Sequencing Consortium (2002) Initial sequencing and comparative analysis of the mouse genome. *Nature* 420:520–562.
- Mural RJ et al. (2002) A comparison of whole-genome shotgun-derived mouse chromosome 16 and the human genome. *Science* 296:1661–1671.
- Overbeek R, Fonstein M, D’Souza M, Pusch GD, Maltsev N (1999) The use of gene clusters to infer functional coupling. *Proceedings of the National Academy of Sciences USA* 96:2896–2901.
- Parkhill J, Wren BW, Thompson NR, Titball RW, Holden MTG, Prentice MB, Sebahia M, James KD, Churcher C, Mungail KL, Baker S, Basham D, Bently SD, Brooks K, Cerdeño-Tárrage AM, Chillingworth T, Cronin A, Davies RM, Davis P, Dougan G, Feltwell T, Hamlin N, Holroyd S, Jagels K, Karlyshev AV, Leather S, Moule S, Oyston PCF, Quall M, Rutherford K, Simmonds M, Skelton J, Stevens K, Whitehead S, Barrell BG (2003) Genome sequence of *Yersinia pestis*, the causative agent of plague. *Nature* 413:523–527.
- Pellegrini M, Marcotte EM, Thompson MJ, Eisenberg D, Yeates TO (1999) Assigning protein functions by comparative genome analysis: Protein phy-

- logenetic profiles. *Proceedings of the National Academy of Sciences USA* 96:4285–4288.
- Reese MG, Hartzell G, Harris NL, Ohler U, Abril JF, Lewis SE (2000) Genome annotation assessment in *Drosophila melanogaster*. *Genome Research* 10:483–501.
- Rogic S, Mackworth AK, Ouellette FBF (2001) Evaluation of gene-finding programs on mammalian sequences. *Genome Research* 11:817–832.
- Rubin GM et al. (2000) Comparative genomics of eukaryotes. *Science* 287:2204–2215.
- Salem A-H, Ray DA, Xing J, Callinan PA, Myers JS, Hedges DJ, Garber RK, Witherspoon DJ, Jorde LB, Batzer MA (2003) Alu elements and hominid phylogenetics. *Proceedings of the National Academy of Sciences USA* 100:12787–12791.
- Simillion C, Vandepoele K, Van Montagu MCE, Zabeau M, Van de Peer Y (2002) The hidden duplication past of *Arabidopsis thaliana*. *Proceedings of the National Academy of Sciences USA* 99:13627–13632.
- Smith NGC, Knight R, Hurst LD (1999) Vertebrate genome evolution: a slow shuffle or a big bang? *BioEssays* 21:697–703.
- Stormo GD (2000) Gene-finding approaches for eukaryotes. *Bioinformatics* 10:394–397.
- Strachan T, Read AP (2003) *Human Molecular Genetics* (3rd edition). New York:Wiley-Liss.
- Tatusov RL, Koonin EV, Lipman DJ (1997) A genomic perspective on protein families. *Science* 278:631–677.
- Tatusov RL, Galperin MY, Natale DA, Koonin EV (2000) The COG database: a tool for genome-scale analysis of protein functions and evolution. *Nucleic Acids Research* 28:33–36.
- Thanaraj TA, Clark F (2001) Human GC-AG alternative intron isoforms with weak donor sites show enriched consensus at acceptor exon positions. *Nucleic Acids Research* 29:2581–2593.
- Van de Peer Y (2004) Computational approaches to unveiling ancient genome duplications. *Nature Reviews Genetics* 5:752–763.
- Wong S, Butler G, Wolfe KH (2002) Gene order evolution and paleopolyploidy in hemiascomycete yeasts. *Proceedings of the National Academy of Sciences USA* 99:9272–9277.
- Zhang MQ (1998) Statistical features of human exons and their flanking regions. *Human Molecular Genetics* 7:919–932.
- Zhang Z, Gerstein M (2003) Of mice and men: Phylogenetic footprinting aids the discovery of regulatory elements. *Journal of Biology* 2:11.

Glossary

Italicized words in the definitions are also glossary entries.

2DE Two-dimensional electrophoresis for proteins, usually accomplished by isoelectric focusing in the first dimension, followed by SDS polyacrylamide gel electrophoresis in the second.

5' to 3' The direction of nucleic acid synthesis such that the 5' phosphate of ribose or deoxyribose is joined to the 3' hydroxyl of the immediately preceding ribose or deoxyribose in a growing RNA or DNA chain. When a single strand is written, the 5' end of the molecule is conventionally written on the left and the 3' end is written on the right.

additive tree A tree for which the distance between leaves can be obtained by adding the weights of the edges on the path between them.

alignment Placing the elements of sequence strings in register to establish degrees of relationship between them. Individual elements will either be aligned as matches or mismatches or will not be aligned opposite another element (indicating an indel).

alignment matrix A matrix whose rows correspond to elements of one sequence string and whose columns correspond to elements of the other in a pairwise alignment. The alignment matrix element m_{ij} contains the best score resulting from aligning the first i elements of one string with the first j elements of the other.

allele One of two or more possible sequence variants for any particular chromosomal locus or gene.

antibody (\equiv immunoglobulin) A protein produced by vertebrate immune systems for binding and inactivating molecules that are not normally found in the body. Also known as immunoglobulins, these molecules occur in a number of different forms having specialized functions.

antibody microarray A two-dimensional grid of individual antibody samples placed at recorded positions on a solid substrate. It is used to identify presence and concentrations of protein species (antigens) in complex protein mixtures.

- antigen** A molecule that elicits an immune response leading to the production of antibodies. Antigens usually differ from normal molecules of the organism, except in autoimmune disorders.
- Archaea** One of the three major domains of life. Archaea (also called Archaeobacteria) are prokaryotes, and in many cases, they are found living in extreme environments (e.g., at high salt or high temperature).
- archaeobacteria** See *Archaea*.
- assembly** The process (or result) of combining short DNA sequence reads derived from a larger molecule or genome to construct the sequence of the complete molecule or genome.
- association analysis** The statistical method that uses correlation between the presence of alleles and disease phenotypes to suggest candidate genes causing those phenotypes.
- attribute** See *predictor variable*.
- autosome** A chromosome other than a sex chromosome. In humans, the autosomes are chromosomes 1 through 22.
- autozygous** A condition in which two alleles are identical because they descended from the same ancestral allele.
- balanced graph** An edge bi-colored graph is balanced if each node has an equal number of edges of each color.
- base-calling** The identification of bases in output from a DNA sequencing reaction, particularly assignment of bases to peaks in graphical output.
- biallelic** The property of genes represented by two different alleles.
- binary search** A process for searching lists that relies on reducing the search space by half at each subsequent step.
- binary tree** A *tree* for which internal nodes have at most two descendant nodes, or *children*. Each internal node of a binary tree splits into two descendant lineages.
- binomial distribution** The probability distribution describing the number of successes and failures in a fixed number of independent trials when only two outcomes are possible.
- BLAST** Basic Local Alignment Search Tool: a method for rapidly searching protein or nucleic acid sequence databases to detect statistically significant local alignments.
- BLOSUM** Block substitution matrices based upon short sequence blocks conserved among different proteins. Used for aligning protein sequences, the various BLOSUMs specify the score to be assigned when amino acid X is aligned opposite amino acid Y.
- Bonferroni correction** A correction to the significance level associated with multiple hypothesis testing, and reflecting the fact that if multiple independent hypotheses are tested for significance as a group, the probability of obtaining a false positive result will increase as the number of hypotheses increases.

- bottom-up** An approach to genomic sequencing that begins with small insert clones and builds the larger physical map by overlapping the maps of the small inserts.
- branch site** The conserved position within intronic sequences to which the 5' end binds during the splicing of mRNA precursors.
- categorical character** A character whose states differ in kind and that cannot be related on a numerical scale. An example would be hair color of mice: white, brown, or grey.
- cDNA** (complementary DNA) A DNA molecule produced by reverse transcription of an RNA molecule. Usually refers to DNA representations of spliced mRNA molecules. See also *library*
- cell cycle** The ordered stages during cell growth, chromosome replication, and cell division. Usually refers to the stages G_1-S-G_2-M for the eukaryotic cell.
- centimorgan** (abbreviated cM) A unit of genetic map distance, such that two markers A and B that are 1 cM apart on the same chromosome are separated by recombination in 1 out of every 100 meioses (i.e., 1% of the time).
- central limit theorem** Sums of independent, identically distributed observations, when properly normalized, have a limiting normal distribution.
- centroid** For a set of m points in n -dimensional space, the centroid is a point in that space such that each of its n components is the average of the m values for that component.
- challenge set** A set of sequences known not to contain the signal of interest. Sometimes produced by a probabilistic model. See *training set, validation set*.
- character** A feature or property used in classification; a property that can differ among OTUs.
- child(ren)** In directed graphs, points connected immediately below or following a vertex are children of that vertex.
- ChIP** See *chromatin immunoprecipitation*.
- χ^2 statistic.** A statistic computed to test the significance of differences between observed values of response variables and those predicted by a specified null model. Distributions of the χ^2 statistic vary with the number of degrees of freedom, which is computed as the number of independent variables minus the number of fitted parameters (if any).
- chloroplast** A membrane-bound organelle in cells of plants and green algae containing the biochemical components for photosynthesis. Sequences of chloroplast DNA molecules suggest a relationship of these organelles to cyanobacteria.
- chromatin** DNA complexed with histones and other chromosomal proteins.
- chromatin immunoprecipitation** (*ChIP*) A method for trapping DNA that specifically interacts with proteins of interest. Bound proteins are chemically cross-linked to DNA and then selectively precipitated by using cognate antibodies.

city-block metric Measure of distance based on steps from point to point parallel to Cartesian coordinate axes in multidimensional space. Sometimes referred to as Manhattan distance.

clade A grouping in a cladogram that includes all taxa descended from any particular node. A clade is defined in terms of character states or features shared by taxa belonging to that clade.

cladogenesis The process of producing clades during evolution.

cladogram A directed graph connecting taxa (OTUs) through shared common ancestors. All taxa (extinct or not) appear at the leaves. Cladograms are hypotheses about relationships among OTUs based upon shared features. Internal nodes correspond to the set of features shared by descendants and not necessarily to an ancestral species.

classification The process of distributing objects into categories based upon the states of characters describing these objects.

clone coverage The number of times that the average base in a genome is represented within cloned inserts in a collection of clones. See *coverage*.

cloning vector A replicating molecule (usually a plasmid, bacteriophage, or artificial chromosome) into which a desired DNA fragment can be inserted for amplification by growth in an appropriate host organism.

cluster of orthologous genes (COG) A set of orthologous genes from different organisms that are (a) each other's best hits in an all-versus-all BLAST search and (b) include at least three organisms for which the criterion in (a) applies for all possible pairs.

clustering The process of grouping together objects, OTUs, or taxa based upon similarities or distances calculated from a number of characters that describe those objects.

coalescent Stochastic model for the evolutionary relationships among DNA fragments sampled from a population.

coding sequence DNA sequence that encodes all or part of a gene product such as a protein or a stable RNA species. Translated exons are examples of coding sequences.

coding strand The strand of duplex DNA containing codons that, in the RNA alphabet, would specify a protein product. Standard genetic code is used, except for substitution of U by T.

codon A trinucleotide (triplet, or 3-word) that specifies an amino acid or stop when parsed by the translation apparatus. The codons in mRNA molecules base pair with the anticodons of the cognate tRNAs during protein synthesis.

codon adaptation index A measure of similarity in codon usage for the coding sequence of a given protein compared with codon usage for highly expressed genes.

COG See *cluster of orthologous genes*.

consensus sequence A short DNA or protein sequence representing a signal, with the letter at each position in the consensus sequence corresponding to the most probable letter at that position.

- conserved segment** A region found in two genomes in which homologous genes retain the same order and relative map position in both genomes.
- conserved syntenly** The co-occurrence of a specified set of homologous genes or loci on a single chromosome in organism X and also on a single chromosome in organism Y. This classical definition is expanded by some investigators to include conservation of gene orders. (Compare with *conserved segment*).
- contig** A gap-free assembly of a larger DNA sequence from smaller, overlapping component sequences. Contigs can be determined from DNA sequences or from restriction mapping data.
- continuous character** A quantitative character whose states can take values from a continuous numerical scale. An example is the cranial volume of fossil hominids. This contrasts with *discrete* characters.
- control** A data set corresponding to the null hypothesis (i.e., that has not been subjected to experimental treatment).
- correlation coefficient** A measure of the dependence between two random variables. The correlation coefficient can range between -1 and $+1$, the extremes arising when one random variable is a linear function of the other. See *covariance*.
- covariance** A measure of the degree to which two variables change in the same direction relative to their respective means. When scaled by the product of the standard deviations of the two variables, the covariance is converted to a *correlation coefficient*.
- covariance matrix** For m random variables X_1, \dots, X_m the covariance matrix is the $m \times m$ matrix of elements c_{ij} that are the covariances of X_i and X_j . The c_{ii} are the *variances* for the X_i s.
- coverage** The number of times that the average base in a genome is represented in a collection of clones. Coverage can be calculated in terms of the entire clone inserts (*clone coverage*) or in terms of sequences actually determined for each clone (*sequence coverage*).
- cycle** In a graph, a sequence of vertex–edge–vertex– \dots that ends at the vertex at which it began.
- cytoskeleton** Structures within eukaryotic cells composed of networks of microtubules, microfilaments, and other molecules. The cytoskeleton helps establish cell shape and architecture and is involved in directed cell movement.
- DDP** See *double digest problem*.
- deletion** An alteration in DNA sequence resulting from removing one or more contiguous bases from the sequence string.
- denature** (DNA or RNA): Disruption of basepairing, thus converting duplex or partially duplex molecules to single-strand forms. (Proteins): Disruption of secondary, tertiary, and quaternary structures to produce polypeptide chains lacking their in vivo (normal) conformations.

dendrogram A graphical representation of the results of hierarchical clustering. OTUs and clusters are joined at positions corresponding to the distances between them.

dichotomous character Any character that can take one of only two possible states. Contrast with *continuous character*.

dideoxy sequencing A DNA sequencing method that employs DNA synthesis to produce products extending from a fixed primer to any of the positions at which the letter being tested can appear. If positions containing A are sought, these terminations are produced by using dideoxy-ATP (ddATP), which lacks a 3'-OH and is therefore not a substrate for further extension.

differential expression The expression of one or more genes to different extents, depending upon growth conditions, treatments applied, or the state of the cell cycle.

diploid The condition of organisms or cells having two copies of each autosome (and hence two copies of each autosomal gene) per somatic cell. Diploid cells contain $2N$ chromosomes, whereas haploid cells contain N chromosomes.

directed graph A graph whose vertices are connected by directed edges.

discrete character A numerical character whose states can take on only a finite number of values. The number of legs of animals or numbers of codons in open reading frames are examples of discrete characters. Contrast with *continuous character*.

dissimilarity A measure of the degree of difference between objects or taxa (OTUs) computed with respect to a specified set of characters.

distance A dissimilarity measure that has the properties of a *metric*. An example is Euclidean distance.

distance matrix For m objects or taxa (OTUs), the $m \times m$ distance matrix D contains elements d_{ij} that are the distances between OTUs i and j .

dominant The property of an allele to confer its phenotype to diploid individuals that are heterozygous for the corresponding gene.

dot matrix An alignment matrix whose elements contain dots for matches and no specification for mismatches. Matching subsequences produce diagonal patterns of dots.

double digest problem (DDP) The problem of determining the restriction map of a DNA molecule based upon fragment sizes produced by two restriction endonucleases employed individually and fragment sizes generated when the molecule is digested with both restriction endonucleases together.

draft sequence A preliminary DNA sequence assembly from data with low sequence coverage and containing unresolved ambiguities.

dye bias Differences in fluorescence intensities of two fluorophores for dye-labeled molecules present at identical concentrations in spotted microarray experiments.

edge That part of a graph connecting two vertices.

- edge disjoint** Two cycles in a graph are said to be edge disjoint if they do not share a common edge.
- edit distance** The number of changes required to transform one sequence string into another. If substitutions are used to effect this transformation, the distance is called the *Hamming distance*. If both substitutions and indels are allowed, the number of changes is called the *Levenshtein distance*.
- electrophoretic mobility-shift assay** (EMSA) An electrophoretic method for identifying DNA fragments that bind to DNA-binding proteins. DNA fragments that are specifically bound produce a band migrating with lower than normal mobility. Also known as gel-shift assay.
- epitope** A particular portion of an antigen that is specifically recognized by an antibody species.
- EST** See *expressed sequence tag*.
- Eubacteria** One of the three domains of life corresponding to typical bacteria found in soils, low-temperature marine and aquatic environments, foods, infections, and intestinal flora. Eubacteria are prokaryotes and are distinct from archaeobacteria, which also are prokaryotes.
- euchromatin** That portion of chromosomes that is less condensed and more transcriptionally active during interphase. DNA in euchromatin can usually be stably maintained in typical cloning vectors. Contrast with *heterochromatin*.
- Euclidean distance** Distances defined by the shortest distance between points in a multidimensional Cartesian space.
- eukaryote** One of the three major domains of life. Eukaryotes are characterized by a true membrane-bound nucleus containing chromosomes complexed with histones, a cytoskeleton, and membrane-bound organelles such as mitochondria and chloroplasts.
- exon** A contiguous segment of DNA that is represented in a processed, mature RNA molecule after splicing has removed intronic sequences. Exon sequences may be translated or untranslated. Information in translated exons appears in polypeptide sequences. Contrast with *intron*.
- expected value** (expectation, mean) A measure of central tendency of a probability distribution or a sample of observations.
- expressed sequence tag** (EST) An EST is part of the cDNA corresponding to a gene. It can be used to identify or locate that gene on a clone or in the chromosome by sequence similarity, hybridization or PCR.
- expression vector** A cloning vector containing a regulatable promoter that initiates transcription directed into the cloned insert, thus allowing expression of genes contained on the insert.
- false discovery rate** In classification, the fraction of those features identified as positives that are in fact false positives.
- FASTA** A rapid local alignment method based upon locations of *k*-words in an alignment matrix. Use of *k*-words produces a sparse matrix, inviting more detailed examination of regions where hits are frequent.
- FDR** See *false discovery rate*

- feature** For microarrays, a feature is a gridded location with defined coordinates, where a specific DNA probe sequence (oligonucleotide, cDNA, or other) has been spotted or synthesized.
- fingerprint** A collection of restriction or PCR fragment sizes derived from and identifying a particular cloned DNA insert or other DNA molecule.
- finished sequence** In a sequencing project, finished sequence specifies the letter at each position of the sequenced region based upon high coverage data with ambiguities resolved by resequencing or other methods.
- fixation** The endpoint of genetic drift, where one allele replaces all others in a population.
- footprint** A DNA region protected by bound proteins from DNA cleavage reagents. It identifies binding sites for sequence-specific binding proteins such as transcription factors.
- gamete** A haploid germ cell, ovum (egg) or sperm, produced by sexual organisms. Fusion of two gametes produces a zygote.
- gene** A genomic locus (or nucleic acid segment) specifying or contributing to a heritable trait associated with an organism. A gene usually codes for a single RNA species or (after translation) a single polypeptide chain.
- gene expression matrix** For n genes whose expression is measured for m conditions, the $n \times m$ matrix of expression levels (typically ratios of treatment to control conditions) is the gene expression matrix.
- gene fusion** The joining of two or more genes as a continuous in-frame DNA sequence encoding a single polypeptide chain containing domains corresponding to each of the two original gene products.
- gene neighbors** Genes mapping close together on the genome. Particularly for organisms whose genes are organized into operons (often prokaryotes), neighboring genes are more likely to be functionally related than genes that are more distant.
- gene pool** The totality of genes contributed by all individuals of a population.
- gene tree** An evolutionary tree constructed by using sequences of a particular gene drawn from different species rather than using species properties. Gene trees and species trees need not be congruent.
- genetic drift** The change in allele frequencies over successive generations of a population arising from sampling effects in the absence of selection, migration, or other specific biological mechanisms.
- genetic map** A diagram for all or part of a genome showing the relative positions of genes or other loci and the distances between them. Distances in genetic maps traditionally are related to frequencies of recombination between pairs of genes, but in current usage may be measured in numbers of base pairs.
- genome** The entire genetic complement of an organism. For eukaryotes in particular, “genome” can refer to the nuclear genome, consisting of the DNA on all of the individual chromosomes. Mitochondrial or chloroplast

genomes are separate genomes characteristic of the organisms that they inhabit.

genomic library A set of cloned fragments representing the entire (usually nuclear) genome of an organism.

genotype The specification of alleles for one or more genes in the genome of an organism. For diploid organisms, the alleles contributed by both parents are listed.

germ line The lineage of cells that will eventually produce gametes. Germ line cells are set aside early in development and may undergo fewer cell divisions than somatic cells.

global alignment Alignment between two sequences such that all letters of both sequences are aligned opposite letters or indels. Contrast with *local alignment*.

global normalization In two-dye microarray experiments, global normalization employs all the features on the array to correct for systematic bias in intensities. This method is based on the assumption that expression under treatment and control conditions is identical for most genes.

graph A set of vertices or nodes along with a set of edges connecting pairs of vertices.

greedy algorithm Method for optimizing the value of a function by taking a sequence of locally optimal steps. The method does not guarantee a global optimum.

Hamiltonian path problem The problem of specifying a path through a graph such that each vertex is visited only once. See *travelling salesman problem*.

Hamming distance The number of substitutions needed to transform one sequence string into another without using indels. Contrasts with *Levenshtein distance*.

haploid The genetic condition of organisms or cells having only one copy of each chromosome per cell. Gametes of diploid sexual organisms are haploid. If the diploid condition corresponds to $2N$ chromosomes, the haploid state corresponds to N chromosomes. Contrasts with *diploid*.

haplotype The combination of alleles of genes on a single chromosome. Contrasts with *genotype*.

haplotype block A segment of genomic DNA that contains markers in linkage disequilibrium within a particular population.

heterochromatin The portion of chromatin that is highly condensed and less commonly transcribed throughout the cell cycle. Heterochromatin often is difficult to maintain stably in cloning vectors. Contrast with *euchromatin*.

heterozygosity A measure of the genetic variation in a population. Often calculated as the chance that two randomly chosen genes at a given locus have different alleles.

heterozygous The condition that results if two different alleles of a gene are present in the diploid state. Contrast with *homozygous*.

hierarchical clustering A process that groups OTUs into successively more inclusive groupings using similarity or distance measures.

hit A database entry matching a query sequence after a database search.

homologs Related genes or loci whose similarity is a consequence of descent from a shared common ancestor. Homologs in different species of organisms are *orthologs*, and homologs within a species are *paralogs*.

homozygous The condition that results if the same alleles of a gene are present in the diploid state. Contrast with *heterozygous*.

hybridization Formation of duplex nucleic acid species during renaturation of single-strand precursors, some fraction of which may be heterologous.

identity An instance of the same letter at an aligned position in two nucleic acid sequence strings.

immunoglobulin See *antibody*.

indel An insertion or deletion of letters applied to either of two sequence strings being aligned.

independent Two events are said to be independent if the probability that they both occur is the product of their individual probabilities. Intuitively, events or random variables are independent if the outcome of one of them is unaffected by knowledge of the outcome of the others.

infinitely many alleles model In population genetics, the approximation that mutations always produce novel alleles.

infinitely many sites model In population genetics, the approximation that mutations always occur at positions in DNA that have not been previously mutated.

information Nonrandomness in a sequence string or signal. This can be measured, for example, by relative entropy.

insertion The addition of one or more nucleotides (or amino acid residues) into a nucleic acid (or protein) sequence.

intensity-dependent normalization Adjustment of data in microarray analysis to compensate for intensity-dependent dye bias.

intron A segment of noncoding DNA separating exons (regions represented in processed mRNA) within genes. Introns are removed by splicing of precursor RNA molecules to form mature mRNAs.

inversion The appearance of a substring of complementary letters in reverse order within an otherwise unchanged nucleic acid sequence string.

inverted repetition In DNA, a second copy of a sequence appearing immediately adjacent or some distance away but on the opposite strand, preserving the proper 5' to 3' polarity. For sequences appearing as inverted repeats, the sequence of letters read left to right on the "top" strand is the same as the sequence of letters read right to left on the "bottom" strand.

island In DNA sequence assembly or physical mapping, an island is a series of overlapping reads or clones representing part of a sequence or physical map. An island can be a single read or clone, or a contig built from several reads or clones.

- isoelectric focusing** A method for resolving a mixture of protein components in solution using an electric field and a stationary pH gradient. Separated protein bands are located at positions in the pH gradient corresponding to their pI (pH for which the net charge on each protein species is zero).
- Jaccard's coefficient** A measure of similarity between OTUs that does not count characters absent from both OTUs.
- K-means** A nonhierarchical clustering method that iteratively places OTUs into a predefined number (k) of clusters, such that the sum of the within-cluster variances is minimized.
- k-tuple** (or *k*-word) A short sequence of k letters.
- lagging strand** The strand of a replicating DNA duplex that is synthesized discontinuously in a direction opposite to the direction of replication fork movement.
- landmark** A DNA sequence used to denote position within the genome. A landmark might be some or all of a gene sequence, or a sequence-tagged site.
- layout** The process of constructing a contig by arranging overlapping sequence reads based on pairwise comparison scores.
- LD** See *linkage disequilibrium*.
- leading strand** The strand of a replicating DNA duplex that is synthesized continuously, with the 5' to 3' direction identical to the direction of replication fork movement.
- leaves** The terminal nodes of a tree together with their edges. Terminal nodes are connected to the tree by a single edge.
- Levenshtein distance** The number of substitutions or indels required to transform one sequence string into another.
- library** A collection of DNA clones whose inserts are a sample of all sequences in a genome or transcriptome. The former collection is called a genomic library, and the latter is called a cDNA library.
- likelihood function** The probability or probability density function of a data set, viewed as a function of the parameters in the underlying statistical model for those data. Used to find *maximum likelihood estimator* of those parameters.
- linkage analysis** The determination of the relative order and positions of genes based upon patterns of coinheritance.
- linkage disequilibrium (LD)** The non-random association of alleles at different loci on a chromosome.
- linked** Genes that tend to be inherited together are said to be genetically linked.
- local alignment** Alignment of substrings taken from each of two different sequence strings. Contrasts with *global alignment*.
- locus** A position on a genetic or genome map defined by the gene or DNA sequence appearing at that position.

- LTR** Long terminal repeat. Repeated retroposon or retrovirus end sequences required for transposition.
- MALDI** Matrix-assisted laser-induced desorption and ionization. A method of generating ionized molecular fragments for analysis by mass spectrometry.
- Markov chain** A probabilistic model for a sequence of dependent random variables. The probability distribution of the next outcome depends on the identity of the k previous outcomes. The case $k = 1$ is often called a one-step Markov chain.
- mate pairs** The DNA sequence reads derived from opposite ends of the same cloned insert. Also called *paired end sequences*.
- maximum likelihood estimator** (MLE) An estimate of a population parameter based on maximizing the *likelihood function* of the data.
- mean** See *expected value*.
- meiosis** The process of cell division and chromosome segregation that produces *haploid* gametes from diploid germ line cells.
- metric** A distance measure that has appropriate properties of symmetry and distinguishability and that satisfies the triangle inequality.
- minimal tiling clone set** A subset of cloned inserts that represents a complete DNA sequence with minimal coverage. Coverage is greater than 1.0 because of overlaps required to recognize adjacent clones.
- mismatch** Nonidentity between two letters, each derived from one of two different sequence strings being aligned or compared.
- mitochondria** (singular: mitochondrion). Membrane-bound organelles in nearly all eukaryotic cells where biochemical processes for oxidative phosphorylation and ATP production occur. Mitochondria, which contain multiple copies of a small DNA genome, are thought to have resulted from a symbiotic relationship established between a eubacterium and an ancestor of eukaryotic cells.
- mitosis** The process of cell division and chromosomal copying that produces two daughter cells, each having the same number of chromosomes as the parent cell. This is the usual mode of propagation for *somatic cells*.
- MLE** See *maximum likelihood estimator*.
- model organism** An organism chosen for extensive biochemical and genetic analysis based upon favorable economics, growth properties, generation times, and similarity to other organisms of interest.
- monoclonal antibody** An antibody derived from a clone that produces only one antibody species recognizing a single epitope of a particular antigen. Contrasts with *polyclonal antibody*.
- motif** A short, conserved local sequence pattern found among a set of proteins or DNA sequences. Motifs are identified on the basis of sequence similarity.
- MS** Mass spectrometry. A method for analyzing charged molecular species accelerated in an electric field and spatially resolved by a magnetic field. Species are distinguished by their differing mass-to-charge ratios, m/z .

- multiple alignment** Alignment of more than two sequence strings. See *pairwise alignment*.
- multiple hypothesis testing** The simultaneous testing of two or more alternative hypotheses.
- mutation** A heritable change in nucleic acid sequence relative to a defined “wild-type” reference sequence. Mutations can include base substitutions, insertions, deletions, and other alterations. Classically, mutations are recognized by alteration of one or more corresponding phenotypes.
- neighborhood sequences** A collection of sequences found from a given sequence by allowing changes in that sequence.
- network** A collection of interactions and dependencies among genes or gene products.
- neutral mutations** Mutations that do not change the phenotype or, for proteins, mutations that do not change the amino acid sequence.
- nodes** Vertices in a graph.
- noncoding sequence** DNA sequence that does not appear in the completed gene products. This includes intergenic sequences, introns, and untranslated regions of exons.
- normalization** (1) Scaling of random variables, often by subtracting the mean and then dividing by the standard deviation, such that their values sum to 1.0. (2) The term used to describe the process of removing systematic variation (such as differences in labelling efficiency between the two dyes in a two-color experiment) from intensity measurements prior to estimating gene expression levels.
- nucleotide diversity** In population genetics, a measure of genetic variation in a population. Often measured as the average pairwise distance among an aligned set of homologous sequences.
- null hypothesis** The standard hypothesis to which a set of data or outcomes are to be compared. The null model is usually conceptually simple, positing a lesser number of assumptions than alternatives being examined.
- object** One of several distinguishable entities under investigation.
- open reading frame** (ORF) A succession of triplet codons uninterrupted by stop codons. There are six possible reading frames corresponding to any duplex DNA sequence, three on the top strand and three on the bottom strand.
- operational taxonomic unit** (OTU) A general term referring to taxa or objects subjected to clustering or evolutionary analysis. Abbreviated OTU, they may be species or populations within species.
- ORF** See *open reading frame*.
- ortholog** *Homologs* appearing in different species. Orthologs normally retain the same function. Contrasts with *paralogs*.
- OTU** See *operational taxonomic unit*.
- outgroup** A more distantly related OTU to which a set of more closely related OTUs is compared during construction of a phylogenetic tree. The

outgroup aids assignment of the ancestral state of the characters employed for building the tree.

p-value The probability of obtaining a more extreme value of a test statistic than that observed in a data set. Used as a statistical measure of the degree of support for a hypothesis.

paired end sequences See *mate pairs*.

pairwise alignment Alignment of two sequence strings. Compare with *multiple alignment*.

palindrome In the general case, palindromes are sequence strings that yield the same letter sequence when parsed left to right or right to left (example: madamimadam). In molecular biology, adjacent inverted repetitions are sometimes called palindromes. For these, adjacent or nearly adjacent DNA sequences on the top and bottom strands are identical when read 5' – 3'.

PAM See *point-accepted mutation*.

paralog A *homolog* arising from gene duplication within a single lineage or species instead of arising by descent in diverging lineages.

parent In a directed graph, a *vertex* immediately above other vertices and connected to them by *edges* is a parent of those vertices, which in turn are called *children*.

PCR See *polymerase chain reaction*.

penetrance The frequency with which the phenotype associated with a gene actually appears among progeny containing that gene. Not equivalent to dominant or recessive properties.

permutation An ordering of a set of objects. Also, the process of generating a different ordering given an initial ordering.

phage display A method for expressing protein binding motifs, particularly the binding regions of antibodies fused to proteins that are components of filamentous bacteriophage coats.

phylogenetic footprint A DNA sequence pattern recognized by its consistent appearance in aligned regions of related genomes.

phylogenetic profiles Vectors for different genes whose elements indicate the presence or absence of these genes in a collection of organisms whose genomes have been completely sequenced. Genes having the same pattern of occurrence in the set of organisms may participate in the same or similar functions.

phylogeny The ancestor-descendent relationships leading to the production of specified organisms during the course of evolution. In contrast with a cladogram, a phylogeny may associate internal nodes with living or extinct ancestral species.

physical map A map of the locations of, and distances between, landmarks on DNA (e.g., genes).

point-accepted mutation (PAM) A set of matrices for scoring amino acid substitutions in alignment of polypeptide sequences. An alternative is the series of BLOSUM matrices.

- point mutation** A mutation changing the identity of a single base in a genome.
- polycistronic** The property of genes so organized that they are transcribed from the same promoter and transcriptionally controlled by the same set of regulatory elements.
- polyclonal antibody** A complex mixture of antibodies recognizing a variety of epitopes of the same antigen and arising from a population of antibody-producing cells. Contrasts with *monoclonal antibody*.
- polygenic trait** A phenotype resulting from the cooperative interaction of two or more genes.
- polymerase chain reaction (PCR)** A method for exponentially amplifying a restricted length of DNA sequence lying between two locations bound by specific primers. Primers bound to opposite strands direct DNA synthesis toward each other so that the intervening sequence is copied. Repeated cycles of denaturation, primer annealing, and DNA synthesis amplify the DNA approximately twofold during each amplification cycle.
- polymorphic marker** A term describing a gene or locus on DNA having two or more alleles, each appearing at a significant frequency in a population.
- polymorphism** The condition of having polymorphic loci or genes.
- population** A collection of interbreeding organisms from the same species occupying the same geographic locale.
- population structure** The collection of subpopulations that together constitute the whole.
- position-specific scoring matrix (PSSM)** A matrix whose rows correspond to letters that occur at positions in a DNA signal or protein motif and whose columns correspond to the positions. Matrix elements are the log-odds scores for each letter at each position, computed relative to an appropriate null model. A PSSM is a particular type of PWM.
- positional weight matrix (PWM)** A probabilistic description of a DNA or protein motif employing a matrix whose rows correspond to the possible letters at each position, and whose columns correspond to positions in that motif. Elements of this matrix are related to the probability of occurrence of each letter at each position. See *PSSM*.
- predictor variable** A variable representing a characteristic of an object and appropriate for describing its properties.
- prefinished sequence** A DNA sequence assembly based on data with the intended degree of sequence coverage but still containing sequence ambiguities requiring resequencing using other clones or primers.
- primer** A relatively short single-strand oligonucleotide that, when base-paired to the complementary region of a DNA or RNA strand, provides the 3'-OH required for elongation by DNA polymerase. Primers are used in PCR, reverse transcription, and dideoxy sequencing reactions.
- principal components analysis (PCA)** A statistical method of data reduction that defines a limited number of predictor variables to account

for a specified large proportion of the variance in the data. The redefined predictor variables are linear combinations of the original variables.

probability density function (probability mass function) A function that is used to calculate the probability of events associated with a *random variable*. The density function is used for continuous random variables, the mass function for discrete ones.

probability distribution A function that gives the probability of observing the possible values of a random variable.

probe For microarray experiments, probes are known sequences emplaced as features on a solid substrate. The probes are used to indicate the presence and amounts of complementary sequences in a sample being tested.

profile A probabilistic description of a protein motif in terms of a PWM.

prokaryote A type of unicellular organism that does not contain a true nucleus, membrane-bound organelles, or complex cytoskeleton. Prokaryotes are usually small (length approximately 1 micron), and some are capable of very short generation times (< 1 hour) under favorable conditions.

promoter A DNA sequence element required for initiation of transcription of a gene, including sites where transcription factors bind to control the time and cell type in which transcription occurs. The core promoter consists of the more limited region where the transcription complex is assembled.

proteome (1) The complete set of all protein species expressed in a particular cell type under a specified set of physiological and environmental conditions. (2) All proteins that can be encoded and produced by a particular genome.

PSSM See *position-specific scoring matrix*.

punctate recombination A recombination model invoking short genomic regions having high frequencies of recombination and separated from each other by long regions having very low frequencies of recombination.

PWM See *positional weight matrix*.

qualitative character See *categorical character*.

quantitative character A character that takes values from a numerical scale.

query A sequence for which matches are sought in database searches.

random variable A variable whose value is determined by the outcome of an experiment in which the outcome is not known ahead of time.

read depth Sequence coverage at any particular position in the genome.

recessive The property of an allele that does not confer its associated phenotype when in the heterozygous condition. Phenotypes in such cases are determined by the dominant allele.

recombination The biochemical process of breaking and rejoining DNA molecules, possibly leading to new combinations of alleles. Recombination in eukaryotes occurs during meiosis.

recombination fraction The probability that alleles at two loci on a chromatid come from different parental chromosomes.

- reductionist approach** The notion that complex biological phenomena can be understood in terms of their components. Compare with *systems biology*.
- relative entropy** A measure of information content that compares the observed probability distribution of outcomes with the distribution for a specified background model. Compare with *Shannon's entropy*.
- repeated sequence** A DNA sequence that appears more than once in a genome. Repeated sequences may be highly repetitive or show lesser degrees of repetition down to simple duplications.
- respiration** The biochemical process for breaking down molecules such as sugars in the presence of oxygen to produce H_2O , CO_2 , and chemical energy in the form of ATP.
- response variable** A variable whose values are determined by the distribution of one or more predictor variables. In statistics, the response variable is often the quantity of most interest to the investigator, and the predictor variables are those variables that influence the value of that quantity.
- restriction endonuclease** (restriction enzyme) An enzyme that cleaves duplex DNA at or near characteristic short (usually palindromic) sequence motifs internal to the ends of the molecule. Type II restriction endonucleases cleave within their respective recognition sequences, which typically range in size from 4 bp to 8 bp. This type of restriction enzyme is particularly useful for genetic engineering.
- restriction fragment length polymorphism** (RFLP) A variation in the presence or absence of a restriction fragment at a particular region of the genome, resulting from the loss or gain of a restriction site.
- restriction map** A physical map of a piece of DNA showing positions of recognition sites of one or more *restriction endonucleases*.
- restriction site** The recognition sequence for binding and directing cleavage by a restriction endonuclease.
- reversal** Given a sequence of elements $X_i, X_{i+1}, \dots, X_{j-1}, X_j$, a reversal is a permutation that presents these elements in reverse order: $X_j, X_{j-1}, \dots, X_{i-1}, X_i$. The elements may be gene blocks, genes or letters in a DNA sequence. An example of the latter case is alteration of $\dots \text{ATGACTGA} \dots$ changing to $\dots \text{ATTCAGGA} \dots$.
- RFLP** See *restriction fragment length polymorphism*.
- root** The node of a tree from which all other nodes descend. The root is characterized by two edges directed away from it and no edges directed toward it.
- SAGE** See *serial analysis of gene expression*.
- scaffold** A construct displaying the relative positions and orientations of sequence contigs as a step toward complete genome sequence assembly. Scaffolds serve as frameworks for completing the assembly.
- scoring matrix** A matrix of rewards and penalties to be applied for aligning different letters opposite one another during sequence alignment. PAM and BLOSUM matrices are specific examples of scoring matrices for proteins.

- search space** A set on which a function is to be optimized. One example is the collection of subsequences in a database being searched for matches to a query sequence.
- segmental duplication** Duplication of one or more genome fractions so additional copies of these genomic segments appear nearby or elsewhere within the genome. Compare with *whole genome duplication*.
- sensitivity** In classification, the fraction of all true “positive” members of a class scored as “positive.” If TP are true positives and FN are false negatives, $\text{sensitivity} = \text{TP}/(\text{TP}+\text{FN})$.
- sequence anchor** A fixed locus on a genome identified by its specific DNA sequence and used for mapping. An STS is one type of sequence anchor.
- sequence coverage** The average number of times a nucleotide in a genome is represented in a collection of DNA sequence reads. For whole-genome sequencing projects using end sequencing of library inserts, sequence coverage is usually less than clone coverage.
- sequence logo** A representation of the probability of occurrence of letters at each position in a signal sequence or *profile* by a stack of letters, each of whose heights is proportional to the frequency of that letter.
- sequence-tagged connector** A DNA sequence (usually an STS) mapping at one end of a cloned insert and used to probe for overlaps with other inserts in a library. Overlapping clones so identified can be merged with it to form contigs.
- sequence-tagged site (STS)** A unique locus on a genomic sequence identified by its DNA sequence, usually associated with a pair of PCR primers that will amplify that site. It is unnecessary for the DNA sequence between the primers to be known a priori.
- serial analysis of gene expression (SAGE)** An open-architecture method for identifying and quantifying expressed sequences by cloning and sequencing concatamers of 3' end sequences derived from mRNA molecules.
- Shannon's entropy** A measure of information content of a signal (e.g., a sequence string) based upon observed probabilities of outcomes compared with a uniform distribution of possible outcomes. Compare with *relative entropy*.
- short interspersed nuclear element (SINE)** A class of transposable DNA elements found in eukaryotic genomes, often at high copy number.
- shotgun sequencing** Determining the sequence of a larger DNA molecule by the assembly of sequences contained in shorter, random sequence reads derived from it.
- signal** A DNA or RNA sequence pattern that specifically binds or interacts with proteins or other macromolecules.
- similarity** (1) The extent to which letters of two different strings match, sometimes expressed as percentage identity. *Homology* usually implies similarity, but similarity does not necessarily indicate homology. (2) In clustering and classification, the extent to which character states describing two OTUs match.

- similarity matrix** A matrix whose entries give a measure of similarity between pairs of objects.
- simple matching coefficient** A measure of similarity between OTUs that includes matches of character states present in both *and* absent in both OTUs. Contrast with *Jaccard's coefficient*.
- simulation** The use of random numbers to investigate the behavior of a stochastic model. Used in many applications in statistics including inference, model fitting and finding the distribution of test statistics. Also, a computational method for generating observations from particular probability distributions.
- SINE** See *short interspersed nuclear element*.
- single-nucleotide polymorphism (SNP)** A polymorphism involving alternative letters at a particular position in a DNA sequence.
- SNP** See *single nucleotide polymorphism*.
- somatic cells** Body cells of multicellular organisms. Somatic cells leave no descendants in the next generation. Contrast with *germ line* cells.
- species tree** An evolutionary tree showing ancestor-descendant relationships among extant and/or extinct species based upon a collection of different characters or genes. Contrast with *gene tree*.
- specificity** In classification, the fraction of all members of a class scored as "positive" that are true "positives." If TP are true positives and FP are false positives, specificity = $TP/(TP+FP)$.
- splicing** Processing of primary RNA transcripts to remove introns and produce mature mRNA molecules containing a continuous coding sequence composed of joined exons.
- spotted microarray** A collection of DNA probes, applied as small aliquots of solutions in a defined pattern onto a solid substrate, used for measuring gene expression levels.
- standard deviation** A measure of the degree of "spread" of a random variable or measured values for a sample drawn from a population. The standard deviation is equal to the square root of the *variance*.
- standardize** To center a series of values relative to the mean and then scale them in units of standard deviation. Resultant values are sometimes represented as the dimensionless quantity *Z*. If the original values are normally distributed, their standardized values will have a distribution that is $N(0,1)$.
- stationary distribution** The distribution of the first observation in a Markov chain that makes the remaining observations have the same distribution.
- stratified** A property of a population that is composed of distinguishable subpopulations.
- STS** See *sequence-tagged site*.
- substitution matrix** A matrix specifying scores to be applied for matching amino acid residues in an alignment. Examples are PAM and BLOSUM matrices.

- supervised** Supervised learning arises in classification problems in which a particular classification is already identified from a training sample, and the investigator wants to predict those classifications in a new testing sample.
- syntenic** Two or more loci on the same chromosome are said to be syntenic.
- syntenic block** A set of two or more syntenic segments with constituent landmark sequences not necessarily in the same order.
- syntenic segment** A set of three or more high-density landmark sequences that exist on a single chromosome and in the same order in two or more species. Similar to *conserved segment*, for which spacing between markers may be greater.
- systems biology** An integrative approach to biology that investigates multiple inputs producing complex biological phenomena. Resulting descriptions often are in terms of networks of interactions between component subsystems. Contrasts with the reductionist approach.
- target** A sequence sought in a background or mixture of other sequences, usually identified by relationship or matches to a query sequence. A target is sought based on prior knowledge that a relationship should exist between it and the query. In database searches, a target is expected to be one of the hits, but not all hits are targets.
- taxon** (plural: taxa) A lineage at a particular level of hierarchical classification for biological organisms. At the animal phylum level, examples of taxa are Mollusca and Arthropoda.
- template strand** The DNA strand that is copied by RNA polymerase during transcription to produce mRNA.
- TOGA** Total gene analysis. An open-architecture approach for gene expression analysis that tags cDNA representations of mRNAs by oligonucleotide sequences. Different species having the same tag are distinguished electrophoretically.
- top-down** An approach to genome analysis that employs large-scale mapping of large clones as a prerequisite to finer-scale mapping and eventual sequencing of smaller clones.
- training set** A set of sequences known to contain the signal of interest. See *challenge set*, *validation set*.
- transcriptome** The complete collection of mature transcripts in a particular cell type under a specified set of physiological and environmental conditions.
- transition** A DNA mutation in which one purine base is replaced by the other (e.g., A→G) or one pyrimidine base is replaced by the other (e.g., C→T). Contrast with *transversion*.
- translocation** A chromosomal or DNA alteration caused by relocating a chromosome segment to another position in the same or a different chromosome.
- transposable element** Any of several types of genomic DNA elements that can propagate themselves by insertion into new locations in the genome.

Some elements encode the transposition functions, but other elements rely on functions supplied by related elements.

- transversion** A DNA mutation in which a purine is substituted by a pyrimidine, and vice versa. Contrast with *transition*.
- travelling salesman problem (TSP)** The problem of specifying edges connecting vertices of a graph such that all vertices appear once in a cycle and the sum of the edge lengths is minimized. Equivalent to *Hamiltonian path problem* with minimization of the path length.
- treatment** A perturbation applied to cells or organisms, such as application of a drug. Can also refer to a selected physiological state compared with a control state.
- tree** A graph of ancestor-descendant relationships among a set of organisms according to a specified evolutionary model. Contrast with a *dendrogram*, which specifies statistical relationships among OTUs without any evolutionary model and whose nodes do not correspond to ancestors.
- triangle inequality** A criterion met by metrics indicating that for any three points a, b , and c , the distance from a to b is less than or equal to the distance from a to c plus the distance from c to b .
- TSP** See *travelling salesman problem*.
- Type I error** In classification or hypothesis testing, rejecting a hypothesis when it is true. Equivalent to a false-negative assignment in classification.
- Type II error** In classification or hypothesis testing, failing to reject a hypothesis when it is false. Equivalent to a false-positive assignment in classification.
- U-unitig** A *unitig* composed of unique sequence DNA, except for possible short stretches at the ends of the unitig.
- ultrametric tree** A *tree* is ultrametric if the distances between any two *leaves* and their shared common ancestor are equal.
- uniform distribution** A probability distribution over an interval $[a, b]$ such that the probability of any subinterval of a given length is identical, regardless of the location of that subinterval in $[a, b]$. The discrete uniform distribution assigns equal probability to each point.
- unique sequence** DNA sequence segments in a genome that occur only once in that genome. Contrast with *repeated sequences*.
- unitig** A short assembly of DNA sequence from a set of DNA sequence reads that contain no contradictory overlaps. The assembly is uncontested.
- unlinked** A property of genes or other genetic markers, that *segregate* during mitosis at frequencies expected for markers on independent chromosomes. After mitosis, unlinked markers appear in the same daughter cell 50% of the time.
- unsupervised** Unsupervised learning arises when an investigator wants to identify structure or clusters in data that may not be readily observable and have not been prespecified. In the clustering context, the clusters are not predetermined.

- validation set** A set of sequences used to test a classification procedure. See *challenge set*, *training set*.
- variable number of tandem repeats** (VNTR) Motif copy number variation at a genomic locus composed of tandem repeats of a motif.
- variance** A measure of the dispersion about the mean of a random variable or a sample of observations, computed as the average of the squared differences between the values and the mean.
- vertex** The position of an object or node to be related to other objects in a graph. Relationships to other objects are indicated by one or more edges connecting the vertex to other vertices.
- VNTR** See *variable number tandem repeat*.
- WGS** See *whole-genome shotgun*.
- whole-genome shotgun** (WGS) A method of genome sequence determination based on assembly of the whole genome from numerous small sequence reads at high coverage without requiring reference to genetic or physical map locations for those reads.
- zygote** The cell resulting from the fusion of two gametes during reproduction of sexual organisms. A zygote gives rise to an embryo after cell division commences.

A

A Brief Introduction to R

A.1 Obtaining R and Documentation

R is a language that is Similar to S (developed at Bell Laboratories) and its commercial implementation S-PLUS (see <http://www.insightful.com/products/splus/default.asp>). R is a software package that provides operation in the interactive mode, powerful statistical analysis tools, and support for good graphics; it also allows programming in a simple way that is a bit like C. Many of the S-PLUS functions and conventions can be used directly with R. If you have a computer account with a university or institute, you may already have access to S-PLUS. Much of what is covered in this appendix applies to S-PLUS as well as R.

R is available for free download: follow the appropriate link at <http://www.r-project.com> to the CRAN (Comprehensive R Archive Network) mirror nearest your location. The current release at the time of writing (1.9.1) is available for the Linux, Microsoft Windows, and Apple Macintosh OSX operating systems. There is extensive documentation. *An Introduction to R* (108 pp.) can be downloaded as a PDF file from the R Project URL. A number of relevant books are listed at the end of this appendix.

The brief introduction in this appendix is designed to get you “up and running” quickly. It does not cover many of the features of R, and you will need to consult one of the other available resources for details. We have tried to write this so that you can still become functional even if you have never done computations interactively from the command line. (The meaning of the last bit will become clear momentarily.)

If you are already familiar with C, you will find that R is like having instant compilation: you can write out functions and perform applications directly without first having to write source code, compile, and then invoke the executable program. However, because of the way R is constructed, an R program and the objects it uses may require more memory than a comparable program written in C. Therefore, you may find that programs written in R will run much more slowly than comparable C programs if the data sets are

large. In this appendix, names of functions, computer code, and output from R will be written in the `courier` font, as is customary.

A.2 First Steps

A.2.1 Starting, Stopping, and Getting Help

After we have downloaded the R package and it has been uncompressed, we can launch the application from the Start menu (Windows) or by double-clicking the icon (Macintosh). A new window called R Console will open, and after a few seconds we should see something like this:

```
R :Copyright 2004, The R Foundation for Statistical Computing
Version 1.9.1 (2004-06-21), ISBN 3-900051-00-3
```

```
R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.
```

```
R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R in publications.
```

```
Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help. Type
'q()' to quit R.
```

```
>
```

The “>” is the R prompt. We type the commands needed for our particular task after this prompt. The command that we have typed in is executed after we hit the Return key. This may take a few moments, and when the operation is complete, the prompt reappears. To quit R, simply type `q()`. We will then be asked if we wish to save our “workspace image”: this contains the set of objects that have been created during this particular session. The result of this differs depending upon the operating system. If we type “y” (for yes), then a pop-up window may open with the workspace image given the name “Rdata.” We can save the workspace image in different directories (folders), each associated with a different project to avoid confusion later.

BAILING OUT: If an R process seems to “hang” (no return of output, no return to the R prompt), the “Escape” key will interrupt the process. We may need to enable the interruption from the Config pull-down window (Macintosh). The Microsoft Windows GUI provides a “stop” icon that also interrupts the process.

Once we have started R, we can get help in two ways. At the R prompt, type `help(function.name)`. This opens a new window that displays text with brief documentation on the particular function (e.g., function `function.name`). For example, type `help(var)` to get information about the variance function named `var`. The same result is obtained by typing `?function.name` after the R prompt. More extensive documentation can be obtained using the default browser. Type `help.start()` after the R prompt. This should launch the browser, and we can navigate to the desired link in the HTML document that appears in the browser window.

A.2.2 Objects

Once R has been launched, it is ready to execute commands immediately without the need to write or compile source code. We can just type things such as

```
> 2*2
[1] 4
>
```

and obtain an answer directly. Unlike C and many other languages, there is no need to declare variables—R knows what they are, depending on how we created them. The `[1]` is a tip-off about R: it is designed to work with vectors, matrices, arrays, and other complicated objects. The result of multiplying 2×2 (using `*` as the multiplication operator) is a scalar quantity composed of one element, and the `[1]` counter indicates that this first element is what is shown.

Let's now create some objects. We do this at the command prompt for now. Later, we indicate how to input and output data to and from R. All variables (scalars, vectors, matrices, etc.) created by R are called **objects**. In C, we assign values to variables (after they have been suitably declared) with the “=” sign. In R, this assignment is done with an arrow: “<-”. If we assign to variable x the value 2×2 , an R object `x` is created.

```
> x<-2*2
```

To view any R object, just type its name, and the contents of that object will be displayed:

```
> x
[1] 4
>
```

WARNING: Don't use an underscore in object names ("`object_name`"). In earlier versions of R and S-PLUS, the underscore is equivalent to `<-`! Don't get in the habit of using the underscore mark as a substitute for `<-`.

To find out what R objects we have *created*, just type `objects()` at the R prompt and a list of them is returned:

```

> objects()
[1] "ckm2" "ckm3" "ckm4" "ckm5" "clone.name"
[6] "i" "last.warning" "syeast.dat" "x" "y"
[11] "yeast.dat"

```

These objects are in the `.RData` subdirectory or workspace within the chosen subdirectory or folder. Under UNIX, it is possible to read R objects using the `more` command, but the resulting gibberish is uninformative: the objects have, in effect, been compiled by R and must be viewed from within the R application by typing their names at the R prompt. If we have large objects (e.g., matrices with thousands of entries), we may want to be careful about checking the sizes of objects first rather than just typing their names. Otherwise we get many screens full of frustrating blur for very large objects (thousands of lines). We can appropriately use the `length` command first to find out how many elements there are in a dimension of an object. This gives an indication of the object size.

A.3 Types of Objects

R provides for a number of different data types: scalars, vectors, matrices, arrays, data frames, and lists. Data frames and lists may include elements such as characters in addition to numeric quantities. In this brief introduction, we only deal with the first four object types with numeric elements. In the last section, we showed how to create a scalar. To make vectors, matrices, and arrays, multiple data entries are required. The `scan()` function provides for this. At this point, we should indicate what the "`()`" means. Functions usually require arguments, which are variables to be operated upon (i.e., objects) that are passed to the function. Some functions (`q()` and `help.start()` are examples) require no arguments. `scan()` as we use it here does not require arguments because we supply them from the keyboard. `scan` can be used with arguments to read in data from external files.

Let's start by making a vector to contain the homework scores of five students for homework set 1. We use the `scan()` function and type in the numbers at the keyboard, hitting Return after each entry:

```

> HW1<-scan()
1: 8
2: 6
3: 9
4: 10
5: 5
6:
>

```

After item 6, the Return key is depressed without making an additional entry. This completes the creation of the object `HW1` and returns the R prompt. To

read the contents of the object that we just created, we only need to type its name:

```
> HW1
[1] 8 6 9 10 5
>
```

We see that a vector has been successfully created. The [1] is the index of the first element of the vector. We can extract any element of the HW1 vector by typing the object name with the index of that element given in square brackets []:

```
> HW1[4]
[1] 10
```

We could also have created this vector by using the “combine” or “concatenate” command `c()`. The result is the same:

```
> HW1<-c(8,6,9,10,5)
> HW1
[1] 8 6 9 10 5
```

Now suppose that we want to enter the grade from four homework sets and one mid-term examination for five students. This information can be stored in a matrix, each row of which corresponds to a particular student. Each column corresponds to the item that was graded (homework set or mid-term examination, for example). We can create a matrix as follows:

```
> grades<-matrix(scan(),byrow=T,ncol=5)
1: 8          14: 8
2: 9          15: 85
3: 7          16: 10
4: 4          17: 9
5: 89         18: 10
6: 6          19: 10
7: 8          20: 90
8: 10         21: 8
9: 10         22: 6
10: 87        23: 5
11: 9         24: 10
12: 7         25: 92
13: 10        26:
Read 25 items
```

Input is printed in two columns to conserve space. We can read the contents of `grades` by just typing its name after the R prompt:

```
> grades
      [,1] [,2] [,3] [,4] [,5]
[1,]    8    9    7    4   89
[2,]    6    8   10   10   87
[3,]    9    7   10    8   85
[4,]   10    9   10   10   90
[5,]    8    6    5   10   92
```

We make a number of comments about this code:

- The `matrix()` function was invoked to tell R to make a matrix, and `scan()` (without an argument) is one of the arguments of the `matrix()` command. `scan()` is used to tell R to accept values entered from the keyboard.
- `byrow=T` says that we are entering the data in rows (true).
- `ncol=5` says that there are to be five columns.
- Note the designation of rows and columns: `[4,]` for row 4, `[,3]` for column 3. The space after or before the comma implies all the column entries for the given row or all the row entries for a given column.

If we want to add names for the rows and columns, we can use the `dimnames()` function together with the `list()` function:

```
> dimnames(grades)<-list(c("Aaron A", "Jones J", "Patel P",
+ "Smith J", "Zhang Q"),c("HW1", "HW2", "HW3", "HW4", "MT1"))
```

We get the result of this operation by just entering the object name:

```
> grades
      HW1 HW2 HW3 HW4 MT1
Aaron A    8    9    7    4   89
Jones J    6    8   10   10   87
Patel P    9    7   10    8   85
Smith J   10    9   10   10   90
Zhang Q    8    6    5   10   92
```

Analysis:

- The character (non-numeric) items are denoted by the quotation marks.
- `c(..., ..., ...)` concatenates or combines the elements in parentheses.
- Note: The `+` in front of `"Smith J"` indicates that the command is wrapping onto the next line. Do NOT include the `+` when typing R commands.
- We can extract any element from this matrix by designating either the dimension names or the row and column numbers: both `grades[4,3]` and `grades["Smith J", "HW3"]` return the same value: 10.

We can join vectors to other vectors or matrices by using the `cbind()` or `rbind()` functions. `cbind()` joins columns, and `rbind()` joins rows. Suppose that a student, Hisako Yamane, transfers into the section whose scores are recorded in `grades`. Her records can be added to a new grade object as follows:

```

> Yamane.H<-c(5,8,10,7,84)
> grades.1<-rbind(grades,Yamane.H)
> grades.1
      HW1 HW2 HW3 HW4 MT1
Aaron A   8  9  7  4  89
Jones J   6  8 10 10  87
Patel P   9  7 10  8  85
Smith J  10  9 10 10  90
Zhang Q   8  6  5 10  92
Yamane.H  5  8 10  7  84

```

Yamane's name can be tidied up using `row.names`:

```

> row.names(grades.1)[6]<- "Yamane H"

```

Note: UNIX and Linux users already know not to employ spaces as separators in file names. If we try to use an object name with a space as separator, R will return a syntax error message. File names that include spaces must be enclosed in quotes " ". As we indicated above, use of “_” is discouraged with earlier versions of R.

We can also select a subset of the data from a matrix. For example, the following command makes a homework matrix:

```

> HW.grades.1<-grades.1[,1:4]
> HW.grades.1
      HW1 HW2 HW3 HW4
Aaron A   8  9  7  4
Jones J   6  8 10 10
Patel P   9  7 10  8
Smith J  10  9 10 10
Zhang Q   8  6  5 10
Yamane H  5  8 10  7

```

This simultaneously constructs a new object, `HW.grades.1`, and extracts only columns 1 through 4 inclusive. (In `[,1:4]`, `1:4` means 1, 2, 3, and 4 inclusive.)

Suppose that data are stored as a table of numbers with row and column labels in a text file called `classlist`. This is not an R object, but we would like to import the data into R. We will have no problems importing data from this text file into R if we created the text file using a UNIX text editor such as `pico`, `vi`, or `emacs`. Microsoft Word documents are not pure text files unless they have been saved as “text only.” We can import data into R by using the `read.table()` function:

```

> grades2<-read.table("classlist")
> grades2
      HW1 HW2 HW3 EX1 PD

```

Merrill	9	0	7	89	y
Lynch	3	6	5	62	n
Pierce	8	8	7	75	y
Fenner	NA	6	2	44	y
Smith	9	10	8	94	n

Here are some useful comments:

- The object created is not a matrix but a data frame since it includes categorical information (last column) in addition to numeric data. We can use `is.data.frame()` to check whether something is a data frame. Not all entries are numeric.
- In this case, `classlist` was a text file in the same working directory (see below). Note the quotes used when this file name is called by `read.table`.
- To be read in as a data frame, the external file needs to have a standard format: the first line of the file should have a name for each variable (i.e., column), and each additional line should start with a row name and be followed by the values of each variable (usually separated by spaces).
- Data frames must have row and column names; these are supplied by R if they are absent. Lines can be skipped from the beginning of a file by using the argument `skip`.

You can also read data from an external file by `scan("filename")`, but the dimension names and categorical information may not be handled gracefully. You may need to edit the input file if you wish to use the `scan()` function for this purpose.

At this point, we should be clear about directories and paths to them. Persons familiar with UNIX, Linux, and MS-DOS are already well-acquainted with this concept. The R application needs to know where to look for external files. We may have launched R from a folder on our desktop. This folder is a subdirectory of our “Desktop” directory. To find out the working directory for R, type the command `getwd()` after the R prompt.

```
> getwd()
[1] "Macintosh HD:Desktop Folder:rm180:"
```

The result is the path name to the subdirectory in which R is operating. In this case, the working directory is the subdirectory `rm180` within the subdirectory `Desktop Folder` of the directory `Macintosh HD`. (With Windows, the result of this command would be something like `C:/Program Files/R/rw1091`.) Suppose that we have a folder `Rwork` in which we are storing individual workspaces (collections of objects created during different working sessions) and their corresponding data files. We may need to change the directory to this folder. You can do this with the `setwd()` command as follows:

```
>setwd("Macintosh HD:Desktop Folder:Rwork:")
```

There are GUI items that do this as well.

Another type of data object is the array. We can think of this as a generalization of the matrix. For example, an array of dimension 3 is a stack (first dimension) of matrices (two dimensions each—rows and columns). An example of an array with three levels can be created by

```
> data<-c(1:24)
> A<-array(data,c(4,2,3))
```

This generates a four-deep stack of 2×3 matrices using `data` (contents 1, 2, ..., 24) to supply the elements. The first level can be viewed as follows:

```
> A[1,,]
      [,1] [,2] [,3]
[1,]    1    9   17
[2,]    5   13   21
```

Try looking at `A[,1,]` and `A[, ,1]` to see cross sections through the stack in each of the three dimensions. Notice the default method of filling the array elements. We have seen how to import data from files using `read.table()` and `scan()`. There are several ways to save data to an external file (i.e., a file that is not an R object). The first is to use the `write.table()` function. Suppose that we wanted to save `grades2` into an external file called `gradelist`. This could be done as follows:

```
> write.table(grades2,"gradelist")
```

The second method is to use the `sink()` function:

```
> sink("gradelist")
> grades2
> sink()
```

The first command created a file called `gradelist` in the working directory and directed all subsequent output to this file—not to the terminal. When `grades2` was entered at the R prompt, the contents of `grades2` were returned, but now to the external file `gradelist` rather than to the screen. The final `sink()` command redirects output to your screen.

To remove objects from the working directory, use `rm()`, supplying the names of the objects to be deleted as arguments:

```
> rm(grades2)
```

deletes the object `grades`.

A.4 Computations

We now show how to perform some simple computations with R using the objects that we created in the previous section. We can calculate the class average for the first examination in column 5 of `grades` using the R function `mean()`:

```
> mean(grades[,5])
[1] 88.60
```

The definition of `mean` is self-evident. `grades[,5]` means that all rows in column 5 will be used to calculate the mean. As we indicated above, if we don't specify a row (e.g., `[2,5]` = row 2, column 5) but leave the row designation blank, all rows are implied. This is computationally useful. For example, suppose that exam 1 scores are out of 105, and we want to state the percentage. We can make a vector of these percentages by the following:

```
> ex1pct<-grades[,5]*100/105
> ex1pct
Aaron A   Jones J   Patel P   Smith J   Zhang Q
84.76190 82.85714 80.95238 85.71429 87.61905
```

Since no rows were specified, the computation was done with all of them, and in this case the result was saved in a new vector.

There is an important and useful notation that allows us to do logical manipulation of matrix elements. Suppose that we decided that anyone who earned more than 90 on the first exam should receive a mid-term grade of A. We can pick out just those elements with the following notation:

```
> ex1<-grades[,5]
> ex1A<-ex1[ex1>90]
> ex1A
Zhang Q
      92
```

The key feature is embodied in the syntax `ex1[ex1>90]`. The logical comparison “greater than” is being performed on all elements in `ex1`.

If each homework is worth 12 points, we can convert the table to a percentage by:

```
> HW.grades.1.pct<-HW.grades.1*100/12
> HW.grades.1.pct
      HW1      HW2      HW3      HW4
Aaron A 66.66667 75.00000 58.33333 33.33333
Jones J 50.00000 66.66667 83.33333 83.33333
Patel P 75.00000 58.33333 83.33333 66.66667
Smith J 83.33333 75.00000 83.33333 83.33333
Zhang Q 66.66667 50.00000 41.66667 83.33333
Yamane H 41.66667 66.66667 83.33333 58.33333
```

Notice how this scalar multiplication operated on all rows. We can find the sum of Aaron's points by applying the `sum()` function to row 1:

```
> sum(grades.1[1,])
[1] 117
```

By using analogous syntax, we can perform computations on elements or groups of elements in matrices, the numeric portion of data frames, and arrays as well.

By the way, we can also perform matrix computations. Suppose that we have the following system of equations:

$$\begin{array}{rcl} 5x & -2y & + z = 15 \\ x & + y & + z = 8 \\ -x & -5y & = 2 \end{array}$$

In matrix notation, this would be represented as

$$\begin{pmatrix} 5 & -2 & 1 \\ 1 & 1 & 1 \\ -1 & -5 & 0 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 15 \\ 8 \\ 2 \end{pmatrix}.$$

Enter the data into a matrix `A` and a vector `v` as described above:

```
> A<-matrix(scan(),ncol=3,byrow=T)
1: 5
2: -2
...
8: -5
9: 6
10:
  Read 9 items
> A
      [,1] [,2] [,3]
[1,]    5  -2   1
[2,]    1   1   1
[3,]   -1  -5   6
```

Notice that we keyed in an incorrect value. We simply correct this as shown below:

```
> A[3,3]<- 0
> A
      [,1] [,2] [,3]
[1,]    5  -2   1
[2,]    1   1   1
[3,]   -1  -5   0
```

Now enter the `v` vector:

```
> v<-c(15,8,2)
```

Invoke the `solve()` function, providing as arguments the matrix of coefficients `A` and the vector of values `v`:

```
> solve(A,v)
[1] 1.2608696 -0.6521739 7.3913043
```

The result is a vector containing the values of x , y , and z . We can check the result by entering the results into one of the equations. Using the first equation, we find:

```
> 5*1.2608696-2*(-0.6521739) + 1*7.3913043
[1] 15
```

By the way, `solve()` will also calculate the inverse of a matrix:

```
> Ai<-solve(A)
> Ai
           [,1]      [,2]      [,3]
[1,]  0.21739130 -0.21739130 -0.1304348
[2,] -0.04347826  0.04347826 -0.1739130
[3,] -0.17391304  1.17391304  0.3043478
```

We also could have solved the system of equations above by using matrix algebra,

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = A^{-1} A \begin{pmatrix} x \\ y \\ z \end{pmatrix} = A^{-1} \begin{pmatrix} 15 \\ 8 \\ 2 \end{pmatrix},$$

where A^{-1} denotes the inverse of A . We can compute x , y , and z by using the appropriate matrix multiplication:

```
> Ai%*%v
           [,1]
[1,]  1.2608696
[2,] -0.6521739
[3,]  7.3913043
```

This is the same answer as we obtained before, now expressed as a 3×1 matrix or column vector.

WARNING: Note the use of the `%*%` operator for matrix multiplication. If we employ the `*` operator, we get an entirely different result. Try it, and observe what happens. Sometimes we may want to use `*`, however.

Note that matrices can also be created from numerical-valued parts of data frames. For example, the command

```
> HWscores <- grades.1[,1:4]
```

extracts columns 1 to 4 from the data frame `grades.1`, and the command

```
> HWscores<-as.matrix(HWscores)
```

makes `HWscores` a matrix. (This can be checked by using `is.matrix(HWscores)`.)

In closing, we note two different operators, `**` and `^`, used to indicate exponentiation:

```
> 9**2
[1] 81
> 9^2
[1] 81
```

A.5 Simple Statistical Applications

R provides powerful tools for statistical analysis. The natural way to see simple data is as a matrix, with each column representing the values of a variable. We illustrate the most basic statistical functions using a data set containing body weights and brain volumes of five primate species. Using a text editor, the data are written into a text file called `primate.dat` in the working directory, and then they are read with `read.table()` into a matrix as indicated above:

```
> primate.data<-read.table("primate.dat")
> primate.data
```

	bodyweight	brainvol
H.sapiens	54000	1350
H.erectus	55000	804
H.habilis	42000	597
A.robustus	36000	502
A.afarensis	37000	384

Calculate the mean and standard deviation (as the square root of the variance) of the brain volume:

```
> mean(primate.data[,2])
[1] 727.4
> sqrt(var(primate.data[,2]))
[1] 380.5362
```

We can use `cor()` to determine the correlation coefficient between `bodyweight` and `brainvol`:

```
> cor(primate.data[,1],primate.data[,2])
[1] 0.828422
```

We can also fit a linear model to the data by using the `lm()` function:

```
>primate.lm<-lm(primate.data[,2]~primate.data[,1])
> primate.lm
```

Call:

```
lm(formula = primate.data[, 2] ~ primate.data[, 1])
```

Coefficients:

```
(Intercept) primate.data[, 1]
-816.29988      0.03446
```

The slope is 0.03446. For more detail, and a tiny indication of what else R can do, issue the command `summary(primate.lm)` after doing the computation above, and see what is returned.

When we start performing computations, output may contain far too many significant digits. The number of digits displayed on output can be controlled by the `digits` argument in `options`:

```
> 22/7
[1] 3.142857
> options(digits=4)
> 22/7
[1] 3.143
> options(digits=7)
> 22/7
[1] 3.142857
```

`round`, `floor`, and `ceiling` are other functions that are useful for controlling data output.

A.6 Functions

A.6.1 Writing Functions

We have been using built-in R functions such as `mean()`, `var()`, `cor()`, `scan()`, and so forth. We can create our own functions in R to perform the tedious calculations. These can be typed out in a text file using a text editor and then, in effect, compiled (converted to R objects) using the `source()` command. This command seeks the file in the working directory, reads it, and then writes the instructions into an executable R function.

Suppose we created the following file, contained in a text file named `xavg.r`:

```
xavg<-function(x)
#calculate the average value of a set of numbers
#expressed as a vector
{
  avg<-sum(x)/length(x)
  return(avg)
}
```

This function will calculate the mean of `x`, where `x` is a vector of numbers. This example duplicates what the supplied R function `mean()` already does. There are a number of points to make about this function:

- “Comment” lines are preceded by the `#` sign. Such lines are ignored by R at execution. As in any programming language, we can use the comments as a way of “commenting” out lines when debugging code. When copying text files back and forth between text editors, it is possible that comment lines in new files may wrap to produce unintended new lines lacking the needed `#` sign. R will complain bitterly about this, and the function will not be created when we `source()` it. Make sure that comments don’t get divided.
- The braces `{}` enclose the body of the function.
- Note the use of the `<-` notation.
- Two of the functions supplied with R are called within this new function: `sum()` and `length()`. `sum()` calculates the sum of the elements contained in the object provided in the argument, and `length()` calculates the number of elements in the object supplied as its argument.
- The `return()` statement returns the calculated value to the workspace. Note that variables defined within functions have local scope: the variable names apply only within the function—new objects are not created in our workspace.

Of course, the code written above is still a text file—it has not been converted to an R object. To do that, we use the `source()` command:

```
> source("xavg.r")
```

If the function has no syntactical errors (e.g., a missing “)” or “}”), the R prompt is returned. The `xavg()` function is now available for use and will be listed as an object in the workspace. To examine the code that makes up a function, just type its name (without any parentheses). Try this by typing `xavg`.

We apply `xavg` to the primate data from the previous section:

```
> xavg(primate.data[,2])
[1] 727.4
```

If `xavg()` had failed to return the same value as was obtained by the function `mean()` in the last section, we would know that we had made a conceptual error (as opposed to a syntactical error) in writing the function. Unlike syntactical errors, conceptual errors do not generate error messages. We can often avoid conceptual errors by testing functions on simple, abbreviated test data for which hand calculation is feasible. `debug(fn)` can be used to debug the function `fn`; it steps through `fn` one line at a time, allowing you to track the values of different variables during function evaluation. `undebug(fn)` turns it off.

A.6.2 Loops

The vector notation of R frequently allows us to circumvent writing loops—a feature called “vectorizing loops.” For example, in Section A.4, we noted that

the code `ex1A<-ex1[ex1>90]` allows us to pick out all items in `ex1` that are greater than 90. Sometimes we may need to write loops, and R allows us to write loops containing a number of statements. The syntax for the `for` loop is illustrated by:

```
for(i in 1:10){
  print(i)
  print(i**2)
}
```

Here are some comments on this code.

- `i` is the counter for the number of cycles through the loop.
- Note that the statements included in the loop are all grouped between `{ }`. This is in addition to the `{}` employed in the function in which this loop might be embedded.
- As written, this loop can be typed in at the R prompt, and it will be executed after the last `}` is entered. It does not have to be located within a function. This means that we can debug an R function in fragments of code.

Another useful command, which can often be faster than loops, is `apply`. For example,

```
> apply(x,1,mean)
```

gives the row mean for each row of the matrix `x`; replacing 1 by 2 gives the column means.

To illustrate the `while` loop, suppose that we create a vector called `result` with ten elements, all initialized to zero. Suppose `integers` is a vector containing integers from 1 through 10.

```
> integers<-c(1:10)      #Make a vector of integers
> result<-c(1:10)       #Make a result vector & initialize
> result[ ]<-0
```

The last two steps can be done in one shot using the `rep` function:

```
> result<-c(rep(0,10))
> result
[1] 0 0 0 0 0 0 0 0 0 0
```

We can run the following bit of code to see how the `while` loop works:

```
> i<-1
> while(i<=5)
+   {
+     result[i]<-integers[i]**2
+     i<-i+1
+   }
> result
[1] 1 4 9 16 25 0 0 0 0 0
```


If we type a return after each line is entered from the terminal, R realizes after it has seen `while` that more commands will be coming, and it prompts us with `+` to continue commands until the end of the loop is reached with the final `}`. We do not type in the `+`.

With the ability to write a series of statements in a function and to perform loops, we can write some reasonably decent programs. The program is called a function in R. Input is handled by supplying one or more arguments to the function (e.g., `xavg(hbrain[,1])` in the example above), and output is achieved by creating an object within the function and returning it using the `return()` statement. We may wish to store the result in an object that we name `tmp`:

```
> tmp<-xavg(integers)
```

We can write `tmp` to a file in the working directory by using `sink()` as described above.

Example: Calculating base composition

The following function calculates the base composition of a DNA sequence string (represented numerically as A->1, C->2, etc.); i.e., it returns the proportion of each type of base. It employs the supplied function `length()` that we have discussed plus the ability to perform logical comparisons (in this case, `==` means “is equal to”) on vector elements, as described in Section A.4.

```
> basecomp<-function(inseq)
#Calculate base composition of an input sequence
#inseq is a vector of integers
{
f<-rep(0,4)           #vector to store base counts
for(i in 1:4)
  {
    f[i]<-length(inseq[inseq==i])/length(inseq)
  }
return(f)
}
```

A.6.3 Libraries

Many authors have written R packages for performing a wide variety of analyses. These may be installed and loaded from within the R GUI. For example, to load the package `MASS`, use

```
> library(MASS)
```

This now makes the routines in `MASS` available in our R session.

A.7 Graphics

Graphics help us grasp complex data at a glance. R allows the production of a variety of plots, including 2-D scatterplots, histograms, piecharts, Boxplots, and perspective plots. For this brief introduction, we indicate only how to do simple two-dimensional plotting and histograms.

A.7.1 Basic Plotting

The first step is to open the graphics device. This is done automatically when we issue the `plot()` command with appropriate arguments or when we set the plotting parameters with the `par()` command. We can close the plotting window either by using the close box on the window or by typing `dev.off()` or `graphics.off()` at the command line. As an example, we plot `brainvol` (y axis) against `bodyweight` (x axis) for `primate.data`.

```
> plot(primate.data[, "bodyweight"], primate.data[, "brainvol"])
```

To set the limits on the axes, we can add arguments to `plot()`:

```
> plot(primate.data[, 1], primate.data[, 2], xlim=c(0, 60000),
+      ylim=c(0, 2000))
```

If we want axis labels, we add more arguments to the `plot()` function:

```
> plot(primate.data[, 1], primate.data[, 2], xlim=c(0, 60000),
+      ylim=c(0, 2000), xlab="Body Weight", ylab="Brain Vol.")
```

Then we add a title:

```
> title("Brain Volumes as a Function of Body Weights for
+      Hominoids")
```

The result is shown in Fig. A.1A.

Sometimes we may wish to preset the plot size and the ranges rather than using the default parameters.

```
> par(pin=c(5, 5))
```

This command sets parameters for the plot area in inches at 5×5 . Plotting symbols other than the default can be used by adding the `pch` argument:

```
plot(x, y, pch=n)
```

where n is an integer from 1 through 18. For example, $n = 1, 2$, or 3 plots the data points as open boxes, open circles, or open triangles, respectively, whereas $n = 15, 16$, or 17 plot data points as filled versions of the symbols above. If we use `pch="A"`, the plotting symbol is the character A. If we want to plot points with connector lines, we issue the command `plot(x, y, type="b")`. For a plot with connector lines only, use the argument `type="l"`.

Suppose that we want to add a horizontal, vertical, or other line to a plot. R supplies the `abline()` function to do this:

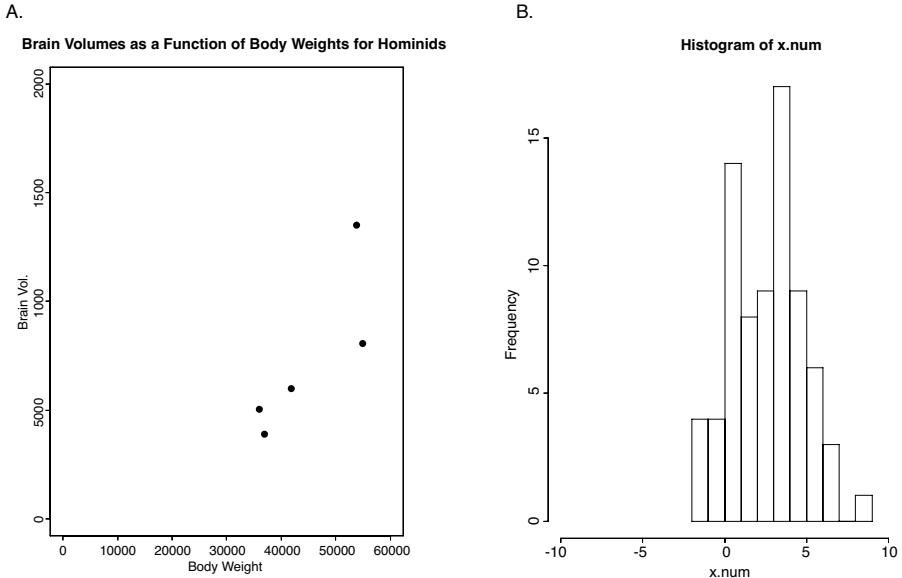


Fig. A.1. Graphical output from `plot` and `hist` functions. For function calls, see the text. Font sizes have been modified for legibility in this book format.

```
abline(h=0)      #Adds horizontal line y = 0
abline(v=3)      #Adds vertical line, x = 3
abline(-0.1,0.2) #Adds line y = 0.2x + (-0.1)
```

Corresponding to `lines()` there is a function `points()`, which adds data points to the plot in the graphics window. For example,

```
>plot(x,y) #Generates scatterplot for data x,y.
>points(u,v,pch=2) #Data points for u, v added with
different print character.
```

When plotting data from different sources on the same plot, we may wish to change the line types for connector lines. This can be done by adding the argument `lty=n`, where $n = 1, 2, \dots$ will generate different types of lines (different sizes of breaks, for example). For example, `points(u,v,type="l",lty=2)` adds to an existing plot additional data as broken lines only.

A.7.2 Histograms

Histograms are another common graphical application. We illustrate this by first generating a set of numbers drawn from a normal distribution to give us something to plot.

```
> x.num<-rnorm(75,mean=2,sd=2)
> hist(x.num)
```

We use the supplied R function `rnorm()`. The first argument is the number of data points to generate, and the second and third arguments set the mean and standard deviation, respectively. The result is shown in Fig. A.1B. We can control the appearance of the histogram by including additional arguments. Often we want to alter the number of classes (`nclass`) or the range of the plot, as in:

```
> hist(x.num,nclass=10,xlim=c(-10,10))
```

`barplot` is useful for plotting from categorical data. `matplot` and `matlines` are useful for plotting rows and columns of a matrix in a single plot. `boxplot` is also useful for summarizing data. We can plot multiple figures (of any type supported by R) on a single display by setting the parameters `mfrow` or `mfcoll`. `mfrow` places the figures on the display sequentially by row, and obviously `mfcoll` does the same thing but by column. Data plotted in the graphics window can be printed or saved in a number of different formats, such as PostScript, PDF, JPEG, or PNG. For example, we can select “Print” from the “File” pull-down menu and then choose “Virtual Printer” and “PostScript Settings” in the pop-up window to write a PostScript file to the chosen destination folder or directory.

References

- Dalgaard P (2002) *Introductory Statistics with R*. New York: Springer-Verlag.
- Krause A, Olson M (2002) *Basics of S-PLUS* (3rd edition). New York: Springer-Verlag.
- Maindonald J, Braun J (2003) *Data Analysis and Graphics Using R*. Cambridge: Cambridge University Press.
- Venables WN, Ripley BD (2002a) *Modern Applied Statistics with S* (4th edition.) New York: Springer-Verlag.
- Venables WN, Ripley BD (2002b) *S Programming*. New York: Springer-Verlag.
- Venables WN, Smith DM, and the R Development Core Team (2002) *An Introduction to R*. Bristol:Network Theory, Ltd.

B

Internet Bioinformatics Resources

A huge amount of genomic and computational biology information is accessible via the Internet. It is impossible to provide a comprehensive overview in the scope of a brief appendix. We therefore focus on key entry points from which to start gathering data. We define three important classes of Web sites: general entry points, databases, and applications. *General entry points* are sites that offer links allowing convenient navigation to a variety of resources. An example is the NCBI Web site (see below). *Database* sites are those that acquire, curate, and distribute information of a specific type, possibly but not always limited to a specific organism. Examples are the *Saccharomyces* Genome Database and InterPro. *Application-oriented* sites offer online access to computational tools. An example of this is the link from the Oak Ridge National Laboratory Web site to the gene recognition tool **GRAIL**.

Reliability and stability are important issues for Internet resources. Almost anyone can post almost anything (that is not blatantly criminal) on a Web site, and URLs that were active yesterday may not be active today. The examples that we provide below are from reputable sources that are likely to exist for a considerable period of time. Other worthy examples have been omitted because of space limitations.

B.1 General Entry Points

B.1.1 National Center for Biotechnology Information (NCBI)

The NCBI Web site at <http://www.ncbi.nlm.nih.gov> is one of the most useful starting points. It also provides links to databases and applications. The NCBI site is an excellent place to begin the search for genomic information, and the numerous links allow many different ways of navigating through the site. The Web site is divided into literature databases, molecular databases, genome biology resources, data-mining tools, learning resources,

and a variety of other more specialized functions, some of which are species-specific (e.g., human-mouse homology maps, rat genome resources, retrovirus resources). We briefly describe just two of these features, molecular databases and data-mining tools, but note that the other sections of this site are extremely information-rich. In particular, the “Books” link provides full-text online access to some of the best biochemistry, cell biology, and genetics textbooks, as well as to a variety of more specialized monographs.

A frequent starting point is a keyword search of one of the available databases: PubMed, Nucleotides, Protein, Online Mendelian Inheritance in Man, or structures, to name a few. Alternatively, **Entrez** can be used in the keyword search to examine the different databases included collectively. Often the purpose is to find and download desired DNA sequences. For example, if the protein database is searched for the string “*topoisomerase I*” *AND archaea*, more than sixty hits are returned, each with an accession number that is hyperlinked to the data file. There are a number of data display options. The default provides sometimes lengthy header information, but if the FASTA format is chosen, the result is one header line followed by the sequence string starting on a new line. In either case, data can be transferred (with the “Send To” button) to a local file or to text, or can be copied and pasted into other applications.

Researchers frequently access the NCBI site to employ the data-mining tools. Examples include various forms of BLAST, Electronic PCR, and ORF finder. The first tool that most people will employ is BLAST, to inquire whether a particular given nucleic acid or peptide sequence displays local sequence similarity to database entries. Nearly every NCBI page has a link to BLAST. Access to the BLAST server is through an HTML-based form that supplies a space to paste the query sequence or search string and menus for selecting databases and setting parameters. The search string can be specified either as FASTA format, a simple character string, or by accession number. Once the search is initiated, seconds to minutes will elapse before the results can be retrieved in a new browser window.

B.1.2 European Bioinformatics Institute (EBI)

The EBI Web site at <http://www.ebi.ac.uk/index.html> is maintained by one of the European Molecular Biology Laboratory’s (EMBL) outstations located in Hinxton, Cambridge, UK. It provides a wealth of information, but most visitors will probably find the “Database” and “Tools” links most useful. Databases include literature databases (e.g., MEDLINE), nucleotide databases (e.g., the EMBL nucleotide database), protein-related databases (e.g., InterPro), macromolecular structure databases, and microarray databases. Much of this information is also available at the NCBI Web site.

The “Toolbox,” which contains additional applications not directly available on the NCBI site, is broken down into five categories: Homology and

Similarity, Protein Functional Analysis, Sequence Analysis, Structural Analysis, and Miscellaneous. For example, the Sequence Analysis category includes pairwise alignment using the Smith-Waterman algorithm, multiple sequence alignment using ClustalW, and alignment of proteins with genomic sequences (GeneWise).

B.1.3 *Science* Magazine Functional Genomics Resources

As with the NCBI and EMBL sites, the *Science* magazine site at <http://www.sciencemag.org/feature/plus/sfg/index.html>, maintained by the American Association for the Advancement of Science (AAAS), contains a wealth of useful links. Most computational researchers will immediately follow the “Scientific Resources” link to find the practical resources, including links to expression data, model organisms, protein structure and function, genome maps, and sequence data. The site does not provide access to computational applications running on AAAS servers; instead, it provides a compendium of useful links to sites that do. The link to “Special Issues” provides subsidiary links to full-text versions of some key genomics articles published in *Science*. This is noteworthy because articles in *Science* are presently otherwise unavailable for download except to members of AAAS.

B.2 Databases

The many bioinformatics databases provide collections of a wide variety of data, including publications, DNA or protein sequences, genes, transcription factors, and protein structures. There are so many databases that it is hard to know where to *stop* looking for data. (We often *start* by using Entrez at the NCBI Web site). Fortunately, a compendium of databases is published annually in the database issue of *Nucleic Acids Research*, for which there is a current link at the *Nucleic Acids Research*/Oxford University Press Web site (Table B.1). Access to the full contents of this particular issue is free. Individual articles in the database issue describe the content and changes of both older and new databases. The 2004 database issue listed 548 databases ranging from AANT (Amino acid-nucleotide interaction database) to ZmDB (*Zea mays* genome database). These two databases also exemplify the wide range of information available. To find a database containing particular information of interest, start with the database issue “Contents” page, follow the link to “Database Listing” in the first article, and then select “Category List.” This provides a set of links that are subcategorized in conceptual hierarchies.

Some databases archive information for individual organisms or for whole collections of organisms (Table B.1). The organism-specific databases may provide the user with different interfaces to the data, and the type of content and organization may differ for each database. Taking TAIR (The *Arabidopsis* Information Resource, Table B.1) as an example, we find access not only

Table B.1. A small sample of useful bioinformatics resources.**General entry points:**

National Center for Biotechnology Information (NCBI):

<http://www.ncbi.nlm.nih.gov>

European Bioinformatics Institute (EBI):

<http://www.ebi.ac.uk/index.html>

Science magazine functional genomics resources:

<http://www.sciencemag.org/feature/plus/sfg/index.html>

Databases:

Nucleic Acids Research (annual Database Issue: link from page below):

<http://nar.oupjournals.org>

a. Collections of organisms

Ensemble (Metazoan animals; e.g., humans, chimps, insects, worms):

<http://www.ensembl.org/>

International Sequencing Consortium (eukaryotic genome sequencing projects):

<http://www.intlgenome.org/>

DNA Data Bank of Japan (DDBJ) Genome Information Broker:

<http://gib.genes.nig.ac.jp/>

The Institute for Genomic Research (TIGR):

<http://www.tigr.org>

UCSC Genome Browser (human, mouse, *Fugu*, yeast, et al.):

<http://genome.ucsc.edu/>

b. Individual organisms

The *Arabidopsis* Information Resource (TAIR):

<http://www.arabidopsis.org>

coliBASE (*E. coli*, *Salmonella*, *Shigella*):

<http://colibase.bham.ac.uk/>

FlyBase (Integrated genomic and genetic data for *Drosophila*):

<http://flybase.bio.indiana.edu/>

MaizeGDB (database for corn genomics):

<http://www.maizegdb.org/>

SGD *Saccharomyces* Genome Database:

<http://www.yeastgenome.org/>

WormBase (data for *C. elegans* and nematodes):

<http://www.wormbase.org/>

c. Molecules and structures

ExPASy (includes links to Swiss-Prot)

<http://www.expasy.org>

InterPro

<http://www.ebi.ac.uk/interpro/>

Table B.1. A small sample of useful bioinformatics resources.**Applications:**

Smith-Waterman alignment (MPsrch at the European Bioinformatics Institute):
<http://www.ebi.ac.uk/MPsrch/index.html>
 GrailEXP (gene discovery, Oak Ridge National Laboratory)
<http://compbio.ornl.gov>
 HMMER (protein sequence analysis using profile hidden Markov models):
<http://hmmer.wustl.edu>

to sequence and gene data but also links to seed stocks and experimental methods. Some databases provide data for a whole variety of organisms that can be viewed with a uniform graphical interface. Examples are the Genome Information Broker of DDBJ (the DNA Data Bank of Japan, Table B.1) and TIGR (The Institute for Genomic Research, Table B.1). Investigators at TIGR pioneered whole-genome sequencing, and their Comprehensive Microbial Resource provides access to genome sequences of more than 130 different microbial genomes and to several eukaryotic parasite and plant genomes. An attractive feature of this site is the different ways of presenting genome information. After specification of an organism, selection of “Analyses” under the “Genomes” tab leads to extensive summary information, such as GC plots, codon usage, and gene categories. If we want information on human and other animal genomes, The Ensemble Genome Browser (provided by the EBI and the Sanger Institute in the UK) or the Genome Browser at the University of California, Santa Cruz, are particularly useful (Table B.1). Databases with listings to multiple organisms often allow searches for features shared among a whole collection of genomes—an obvious benefit to investigators interested in comparative genomics. The International Sequencing Consortium Large-scale Sequencing Project Database provides an overview of all eukaryotic sequencing projects and also supplies links to the sites described above.

Other important databases provide data on molecules and molecular structures. A good example is the ExPASy (Expert Protein Analysis System) proteomics server maintained by the Swiss Institute of Bioinformatics (Table B.1). Here are found not only links to the venerable Swiss-Prot database but also access to three-dimensional views of molecules and to a number of sequence analysis tools. An alternative route to structure information is through the NCBI Web site: select “Structures” and navigate to the Molecular Modeling Data Base (MMDB). After we search using a particular search string, the files returned can be displayed by using an appropriate “viewer” (e.g., Cn3D, RasMol, or MAGE), which can be downloaded for running at local terminals. For example, Cn3D displays the structures in three dimensions, and the structures can be rotated by clicking on the structure and moving the mouse. Note that commercial vendors may also provide

access to useful molecular information. For example, New England Biolabs (<http://www.neb.com/nebecomm/default.asp>) provides extensive information about restriction endonucleases (cleavage sites and reaction conditions) and common cloning vectors (sequences and restriction digest maps).

With the proliferation of databases, each with a potentially different presentation of content to users, databases that integrate content from several different databases are particularly useful. An excellent example of this is InterPro (Table B.1). Maintained at the European Bioinformatics Institute, InterPro integrates information from the UniProt, PROSITE, Pfam, PRINTS, ProDom, SMART, TIGRFAMS, PIRSF, and SUPERFAMILY databases and provides different descriptions of structure patterns, motifs, and domains for proteins of interest.

B.3 Applications

Many bioinformatics application programs are available for download or for online use at no charge. For example, the Smith-Waterman alignment program can be implemented online from the European Bioinformatics Institute Web site as MPsrch (see Section B.1.2) using their server. The “Tools and software packages” link at the ExpASy site provides a link to MPsrch.

Other specific applications may be distributed less widely. For example, GRAIL (Gene Recognition and Assembly Internet Link) and the extended GrailEXP (Grail Experimental Gene Discovery Suite) are implemented mainly by Oak Ridge National Laboratory (Table B.1). Applications at a particular site may be available only for download and are not run on the host server. HMMER (profile hidden Markov models for analyzing protein sequences), provided from the site listed in Table B.1, is an example of such an application.

Some services are freely available from for-profit organizations. For example, the IBM Bioinformatics and Pattern Discovery Group provides access to a number of applications, including tandem repeat discovery, multiple sequence alignment, and gene identification (<http://www.research.ibm.com/bioinformatics>).

B.4 Concluding Remarks

Internet resources in bioinformatics are vast. There are resources other than those listed here that also would have been excellent examples. We have presented several types of reliable resources from a number of different countries and kinds of organizations (academic, governmental, nonprofit institute, industrial). Using the links provided at these sites and Internet search engines, you will be able to find and bookmark your own favorite list of useful bioinformatics sites tailored to your specific needs and interests. For a more comprehensive treatment, see, for example, Baxevanis and Ouellette (2001).

References

- Baxevanis AD, Ouellette BFF (2001) *Bioinformatics: A Practical Guide to the Analysis of Genes and Proteins* (2nd edition) New York: Wiley-Interscience.

C

Miscellaneous Data

C.1 IUPAC-IUB Symbols

Table C.1. IUPAC-IUB symbols for amino acids and combinations of bases.

Amino acid residues:

Alanine	Ala	A	Leucine	Leu	L
Arginine	Arg	R	Lysine	Lys	K
Asparagine	Asn	N	Methionine	Met	M
Aspartic acid	Asp	D	Phenylalanine	Phe	F
Cysteine	Cys	C	Proline	Pro	P
Glutamine	Gln	Q	Serine	Ser	S
Glutamic acid	Glu	E	Threonine	Thr	T
Glycine	Gly	G	Tryptophan	Trp	W
Histidine	His	H	Tyrosine	Tyr	Y
Isoleucine	Ile	I	Valine	Val	V
Unspecified	Xaa	X			

Bases and base combinations in nucleic acids:

A = Adenine	R = A or G (purine)	M = A or C
C = Cytosine	Y = T or C (pyrimidine)	B = T, G, or C
G = Guanine	S = G or C	V = A, G, or C
T = Thymine	W = A or T	H = A, T, or C
U = Uracil	K = G or T	D = A, T, or G
	N = any base	

C.2 Genetic Code

Amino acid residues encoded by each codon are tabulated using the code as it appears in DNA. For RNA code, substitute U for T. Each codon is formed by concatenating three letters taken from successive levels in the tree.

Position			Amino acid	Position			Amino acid
1	2	3		1	2	3	
A	A	A	Lys	G	A	A	Glu
		C	Asn			C	Asp
		G	Lys			G	Glu
		T	Asn			T	Asp
	C	A	Thr		C	A	Ala
		C	Thr			C	Ala
		G	Thr			G	Ala
		T	Thr			T	Ala
	G	A	Arg		G	A	Gly
		C	Ser			C	Gly
		G	Arg			G	Gly
		T	Ser			T	Gly
T	A	Ile	T	A	Val		
	C	Ile		C	Val		
	G	Met		G	Val		
	T	Ile		T	Val		
C	A	A	Gln	T	A	A	STOP*
		C	His			C	Tyr
		G	Gln			G	STOP*
		T	His			T	Tyr
	C	A	Pro		C	A	Ser
		C	Pro			C	Ser
		G	Pro			G	Ser
		T	Pro			T	Ser
	G	A	Arg		G	A	STOP*
		C	Arg			C	Cys
		G	Arg			G	Trp
		T	Arg			T	Cys
T	A	Leu	T	A	Leu		
	C	Leu		C	Phe		
	G	Leu		G	Leu		
	T	Leu		T	Phe		

C.3 *E. coli* Promoter Sequences

Table C.2. A collection of promoter sequences from *E. coli*. These sequences have been aligned relative to the transcriptional start site at position +1. Sequences from -40 to +11 are shown. Close matches to consensus -35 and -10 hexamers are underlined in the first nine entries, and the +1 position is indicated in boldface (Hershberg et al., 2001; <http://bioinfo.md.huji.ac.il/marg/promec>).

	-35	-10	-1
ORF83P1			
<i>ada</i>	CTCTGCTGGCATT <u>CACAA</u> ATGCGCAGGGG <u>AAAA</u> CGTTTCCTGTAGCACCG		
<i>amnP4</i>	GTTGGTTTTTGCGTGATGGTGACCGGCAGCCTAAAGGCTA TCCTTAACCA		
<i>araFGH</i>	TTCACATTTCTG <u>TGACA</u> TACTATCGGATGTGCGGTAATTG TATGGAACAGG		
<i>aroG</i>	CTCTCCTATGGAGAATTAATTTCTCGCT <u>AAAA</u> CTATGTCA A CACAGTCACT		
<i>atpI</i>	CCCCGTT <u>TACA</u> CATTCTGACGGAAGATATAGATTGGAAGT A TTGCATTAC		
<i>caiT</i>	TATTGTTT <u>GAAA</u> TCACGGGGCGCACCGT <u>TATAA</u> TTGACCGCTTTTTGATG		
<i>clpAP1</i>	AATCACAGAATACAGCTTATTGAATACCC <u>ATTAT</u> GAGTTA G CCATTAACGC		
<i>crrP2-I</i>	TTAT <u>TGACG</u> TGTACAAAAATCTTTTCTTATGATGTAGAA G CGTGCAACGC		
<i>cynT</i>	GTGGTGAGCTTGTGCGGATGAACGTGCT <u>TACACT</u> TCTGTTGCTGGGGATGG		
<i>damP2</i>	GACTTTTACCTTATGACAATCGGCGAGTAGTCTGCCTCTCA TTCCAGAGAC		
<i>dnaAP1</i>	TCAGTTGCCAAACCCGCTGGAGTATTGAGATAATTTT CAGT CTGACTCTCG		
<i>dnaQP2</i>	ATCGTGCCCGCTCGCGGCAGGATCGTTTACACTTAGCGA G TTCTGGAAAG		
<i>fabA</i>	GTGAAAAATTTCTACCTGTTAAGCATCTCTGGTAGACTT CT GTAAATTGAA		
<i>fimBP2</i>	TCGGACTTGTT CAGCGTACACG TGTAGCTATCCTGCGT G CTTCAATAAAA		
<i>ftsJ</i>	CTGAATTTTTTATGTTGATTTTACTTGT TACAGA ACATAT C ACATGATATA		
<i>furPb</i>	TGGGATTGAAAACGGGTCATTCTACCGCCATCTCCCATATA TCACCA ATA		
<i>gapB</i>	GGGACTTGTGGTTTT CATTTAGG CGTGGCAATTCTATAAT G ATACGCATTA		
	AAACATTCCTTTTATTCCACGTTTCGCTTATCTAGCTGAA G CGTTTCAGT		

Table C.2. [continued]

	-35	-10	-1
<i>glnAP1</i>			
	GTCATTGCACCAACATGGTGCTTAATGTTTCCATTGAAAGCA	ACTATATTGGT	
<i>glpD</i>			
	AATGTTACCTAAAGCGGATTCTTTGCTAATATGTTTCGATA	ACGAACATTT	
<i>gnd</i>			
	GCTATTTATACTTTAATAAGTACTTTGTATACTTATTTGCG	AACATTCCAG	
<i>hisA</i>			
	AATTAATAAATAGTTAATTAACGCTCATCATTGTACAATGA	ACTGTACAAA	
<i>hypB</i>			
	TTGCCGCGGCAGCGTGGCGGAAGGTTGTAAACTGCACCTC	G AAGAACAAGA	
<i>ilvIHP</i>			
	TGCCAATTGCTTAAGCAAGATCGGACGGTTAATGTGTTTT	A CACATTTTTT	
<i>lep</i>			
	CTCAATGTTGTAGTGTAGAATGCGGCGTTTCTATTAATACA	GACGTTAAGC	
<i>leu</i>			
	AGGGTTGACATCCGTTTTTGTATCCAGTAACTCTAAAAGCA	TATCGCATT	
<i>lysUP2</i>			
	GAGGTAAGCGTTAGTTTCGATAAGATAAACTGAGTTACTAA	TAGTCGAGGC	
<i>melA</i>			
	GCCGGAGGTTTTCTGCAGATTCGCCTGCCATGATGAAGTT	ATTC AAGCAAG	
<i>melR</i>			
	ACTCGCAGTCATCCTCCCTCACTCCTGCCATAATTCTGAT	ATTC CAGGAAA	
<i>nagBP1</i>			
	GCTTAAAGATGCCTAATCCGCCAACGGCTTACATTTTACT	TATTGAGGTGA	
<i>nrd</i>			
	ATGCACTTGCAAGAGGGTCATTTTCACACTATCTTGCACT	GAATCCCAAAC	
<i>osmBP1</i>			
	GGCAAATCATCCGCTCTAAGATGATTCCCTGGTTGATAATT	AAGACTATTTA	
<i>PdhR</i>			
	CTGTATGGACATAAGGTGAATACTTTGTTACTTTAGCGTCA	CAGACATGAA	
<i>pfIP6</i>			
	TATCAATTTCTCATCTATAATGCTTTGTTAGTATCTCGT	C GCGACTTAAT	
<i>phoS</i>			
	TTACATATAACTGTCACCTGTTTGTCTATTTTGTCTCTC	G TAGCCAACAA	
<i>prs</i>			
	GAGGTTGATGCGGTGCTTTCCTGGCTGTTAGAATACGCCC	CGTCGCGCCTG	
<i>purFP2</i>			
	GAATGCGCCCCGAACAGGATGACAGGGCGTAAAATCGTGG	G ACACATATGG	
<i>pyrBP1</i>			
	ACTCCGCCCTATAAGTCGGATGAATGGAATAAAATGCATA	TCTGATTGCGT	
<i>recA</i>			
	AAACACTTGATACTGTATGAGCATACAGTATAATTGCTTCA	ACAGAACATA	

Table C.2. [continued]

	-35	-10	-1
<i>rmh</i>			
	AATTGCAGTGCTCATAGCGGTCATTTATGTCAGACTTGTCTTTTACAGTT		
<i>rpoDPhs</i>	CACCCTTGAAAACTGTCGATGTGGGACGATATAGCAGATAAGAATATTGC		
<i>rpsUP2</i>	GCTTTACAAAGCAGCAGCAATTGCAGTAAAATTCCGCACCAATTTGAAATA		
<i>sodA</i>	GATAATCATTTTCAATATCATTAAATTAATACTATAATGAACCAACTGCTTAC		
<i>ssbP1</i>	AGTATTGGAATGCATTACCCGGAGTGTGTGTAAACAATGCTGGCCAGGTT		
<i>tau</i>	AATTCTTTTAAATGAATGTTTTTATTCCTGAATACTGCTCCATAACAAGAC		
<i>treB</i>	TCCCGTTTTTAAATTTTTCCGCGCAATATATTCTGCAGCCAACCAAAAATG		
<i>tufB</i>	TTAGTGCATGAACCTGCATGTCTCCATAGAATGCGCGCTACTTGATGCCG		
<i>umu</i>	ATCAGTATTGATCTGCTGGCAAGAACAGACTACTGTATATAAAAACAGTAT		
<i>wvrCP3</i>	CAGTTTGTCTGAACGTGAATTGCAGATTATGCTGATGATCACCAAGGGCCA		

C.4 Yeast Gene Expression over Two Cell Cycles

mRNA was extracted from yeast cells at ten-minute intervals after reinitiation of the cell cycle and was converted to cDNA. The cDNA from each time point was hybridized to oligonucleotide arrays and then labeled with a fluorophore. The data entries below are fluorescence intensities for a single dye at each time point (Cho et al., 1998). The first entry for each vector is the yeast ORF name.

YBL023c 60.0 135.0 176.0 91.0 67.0 99.0 143.0 238.0 260.0
161.0 124.0 113.0 72.0 113.0 140.0 163.0

YBL072c 2422.0 3077.0 4824.0 2810.0 4349.0 2241.0 3858.0
2948.0 2166.0 5325.0 4145.0 2527.0 4755.0 5332.0 5043.0 5300.0

YBR202w 114.0 273.0 179.0 125.0 115.0 145.0 289.0 491.0
478.0 369.0 218.0 187.0 176.0 295.0 483.0 411.0

YDR258c 714.0 322.0 49.0 60.0 45.0 60.0 27.0 54.0 65.0
74.0 78.0 80.0 69.0 82.0 129.0 143.0

YEL032w 180.0 257.0 184.0 169.0 153.0 146.0 216.0 337.0
276.0 208.0 164.0 164.0 143.0 134.0 222.0 232.0

YER131w 1968.0 2104.0 2719.0 2919.0 3276.0 2260.0 2804.0
2252.0 1721.0 2458.0 2591.0 2261.0 2762.0 1892.0 3040.0
3111.0

YGL189C 3347.0 3055.0 4425.0 4592.0 3955.0 3215.0 4154.0
3204.0 2558.0 3744.0 3764.0 3094.0 3440.0 3268.0 4622.0
4116.0

YGR027C 493.0 775.0 951.0 826.0 721.0 640.0 927.0 707.0
650.0 983.0 1108.0 746.0 933.0 1122.0 1191.0 1074.0

YLL026w 756.0 325.0 143.0 126.0 140.0 101.0 117.0 109.0
182.0 154.0 235.0 208.0 143.0 168.0 257.0 341.0

YLR259C 2038.0 1466.0 1444.0 1004.0 1194.0 982.0 1103.0
1227.0 968.0 1301.0 1197.0 1188.0 1400.0 1491.0 1615.0
1411.0

YLR274W 209.0 262.0 231.0 140.0 148.0 165.0 184.0 271.0
416.0 351.0 243.0 198.0 162.0 179.0 252.0 324.0

Identifications of ORFs:

Heat shock genes	Members of the MCM complex	Ribosomal protein genes
YDR258c: HSP78	YBL023c: MCM2	YBL072c: RPS8A
YLL026w: HSP104	YEL032w: MCM3	YER131w: RPS26B
YLR259C: HSP60	YLR274W: CDC46	YGL189C: RPS26A
YPL240C: HSP82	YBR202w: CDC47	YGR027C: RPS31A

C.5 Preprocessing of Microarray Data

This section gives a brief overview of how to preprocess microarray data in R.

Step 1: Read in the data

The original data are in a spreadsheet that displays the output from the array reader (downloadable from <http://www.cmb.usc.edu>). Using the spreadsheet application program, we collect together (a) the column of identifiers, (b)

and c) the columns for background-subtracted median values for each wavelength, and (d) the column of flags (four columns total). This is saved as a tab-delimited text file. If we parse down the identifier column (first column), we see that some identifiers contain spaces and characters that will be read as R operators. This creates problems importing the data into R. Remove all spaces from the tab-delimited file, and then convert all tabs to spaces. Failure to remove spaces in the identifier column yields the following result when we try to use `read.table()`:

```
>array.dat<-read.table("array_ex.txt_s",row.names=TRUE)
Error in scan(file = file, what = what, sep = sep,
             quote = quote, dec = dec,):
  line 3361 did not have 4 elements
```

It is tedious to remove various other non-alphanumeric characters (e.g., `-`, `"`, `'`, `(`, `)`) so that `read.table()` will work. Eventually, we may need to use:

```
> array.dat<-read.table("array_ex.txt_s",as.is=TRUE)
> array.dat[1:5,]
      V1    V2    V3  V4
1 GH01040 19404 6040  0
2 GH01059 12628 3352  0
3 GH01066  2236  893  0
4 GH01085   474  801  0
5 GH01088   820 1062  0
```

Without the `as.is=TRUE` argument, you may get the following error message:

```
Error in read.table("array_ex.txt_s", row.names = TRUE) :
  invalid row.names specification
```

The identifier column has experimental meaning to the biologist and reflects idiosyncracies of biological nomenclature. Remove the offending characters without complaining.

Step 2: Remove flagged data

The flags are in the fourth column. Nonzero values indicate rows that should be eliminated because the data for that feature are unreliable. We create an object called `array.dat.r` and then use the code below to extract the appropriate rows. The original data set contains 9216 rows, so the process may take several minutes to complete.

```
> array.dat.r<-array.dat[1,]
> array.dat.r
      V1    V2    V3  V4
1 GH01040 19404 6040  0
> for(i in 2:length(array.dat[,1])){
```

```

+ if(array.dat[i,4]==0)
+ array.dat.r<-rbind(array.dat.r,array.dat[i,])
+ }
#Two lines above specify inclusion of a row only if
#its flag is equal to zero.
> length(array.dat.r[,1])
[1] 5641 #Number of rows not flagged

```

We see that only 5641 rows out of 9216 are not flagged. (The original data had a “bad” block, and many rows near the end contained zero for all intensity values.)

Step 3: Calculate M and A, and obtain the factor for global normalization

We make vectors of appropriate length and then compute the appropriate logarithms in base 2:

```

> M<-rep(0,5641)
> M[]<-log2(array.dat.r[,2]/array.dat.r[,3])
> A<-rep(0,5641)
> A[]<-log2((array.dat.r[,2]*array.dat.r[,3])**0.5)

```

At this point, we can compute the factor needed for global normalization. Remember that we found that $\log_2(R/G) = \log_2 k$, so we seek the mean value of M :

```

> mean(M[])
[1] NaN

```

The “NaN” indicates a number not available. To find out where the problem lies, we hunt for the offending row in M and check in A as well:

```

> (1:length(M))[M=="NaN"]
[1] 2096
> (1:length(A))[A=="NaN"]
[1] 2096

```

Only row number 2096 is problematic. We look in `array.dat.r` to identify the problem:

```

> array.dat.r[2096,]
      V1 V2  V3 V4
2292 string -9 2968 0

```

Obviously, we can’t take the logarithm or square root of a negative number; therefore, we need to exclude this row. (Note that the listed row number of the original matrix is not the same as the actual row number after the flagged rows were removed.)

```

> a<-A[c(1:2095,2097:5641)] #Include all but row 2096
> m<-M[c(1:2095,2097:5641)]

```

Now we can find the global normalization factor:

```
> mean(m[])
[1] 0.2903073
> 2^mean(m[])
[1] 1.222901
```

This means that for all spots taken as a whole, R is 1.22 times greater than G . Normalization of this type would correct the R values by multiplying by $1/1.22$, or 0.8177 .

Remove row 2096 from `array.dat.r`, and finally combine the data into a larger matrix.

```
> array.dat.r2<-array.dat.r[c(1:2095,2097:5641),]
> array.a.m<-cbind(array.dat.r2,a,m)
> array.a.m[1:10,]
      V1    V2    V3 V4      a      m
1 GH01040 19404 6040 0 13.402200 1.6837336
2 GH01059 12628 3352 0 12.667572 1.9135321
3 GH01066  2236  893 0 10.464610 1.3241881
4 GH01085   474  801 0  9.267201 -0.7569152
5 GH01088   820 1062 0  9.866024 -0.3730880
6 GH01132  3054 2032 0 11.282585  0.5877997
7 GH01314  3104 1507 0 11.078688  1.0424491
8 GH01331   479  683 0  9.159812 -0.5118599
9 GH01338  2469  784 0 10.442210  1.6550013
10 GH01353   903  560 0  9.473933  0.6892992
```

This matrix (5640×6) contains the data necessary for generating the MA plot.

A text version of this file is available for download at <http://www.cmb.usc.edu>.

References

- Cho RJ, Campbell MJ, Winzeler EA, Steinmetz L, Conway A, Wodicka L, Wolfsberg TG, Gabrielian AE, Landsman D, Lockhart DJ, Davis RW (1998) A genome-wide transcriptional analysis of the mitotic cell cycle. *Molecular Cell* 2:65–73.
- Hershberg R, Bejerano G, Santos-Zavaleta A, Margalit H (2001) PromEC: An updated database of *Escherichia coli* promoters with experimentally identified transcriptional start sites. *Nucleic Acids Research* 29:277.

Index

- ** , 490
- # , 492
- %*% , 490
- ^ , 490
- 2D gel electrophoresis, 328
- 2DE, *see* gel electrophoresis, 328
- 3' acceptor site, 435
- 3' to 5' , 20
- 5' donor site, 435
- 5' to 3' , 20

- Ab, *see* antibody
- abline**, 496
- abundance, 327
- acceptor group, 228
- accession number, 500
- adapter, 296
- additive, 350
 - distance, 352
 - tree, 351, 352
- adjacency, 127
- affinity tag, 328
- agglomerative
 - clustering, 272
 - hierarchical clustering, 272
- algorithm, 105
 - rejection, 405
- alignment, 144, 167
 - global, 145, 147, 152, 163
 - highest-scoring, 146
 - local, 145, 155, 156, 163
 - matrix, 148, 153, 157, 173, 204
 - multiple, 161, 205
 - multiple-sequence, 145
 - number of, 159
 - pairwise local, 163
 - parsimonious, 145
 - protein, 184
 - score, 153
- all
 - against-all, 443
 - versus-all, 451
- allele, 9, 15, 362, 368, 394
 - frequency, 367, 368, 370, 374
 - new, 368
- allelic association, 386
- alphabet, 38, 70
- alternating cycle, 133
- alternative
 - splice variants, 28
 - splicing, 17
- Alu element, 380, 417
- amino
 - acid, 6
 - acid residue, 21
 - terminal, 21
- amplifying DNA, 67
- ancestor-descendant relationship, 337
- ancestral
 - node, 348
 - state, 340, 349
- anchor sequence, 424, 427
- anchoring enzyme, 294, 296
- animal, 2, 448
 - cell, 6
 - phylogeny, 100
- annotate, 313

- annotation, 189, 443
- anonymous gene, 313
- antibody, 32, 227, 329
 - microarray, 329
- antigen, 32, 329
 - capture, 330
- antiparallel, 20
- apparent island, 113
- apply, 494
- Arabidopsis thaliana*, 4
- ARACHNE, 210
- Archaea, 2
- archaeobacteria, 2
- architecture
 - closed, 292, 298
 - open, 292, 327
- argument, 482
- array, 90, 242, 487
- array, 90, 487
- artifact, 302
- as.dist, 275
- as.is, 513
- as.matrix, 490
- assembler
 - ARACHNE, 210
 - CAP, 210
 - EULER, 210
 - Phrap, 210
 - TIGR, 210
 - Whole-genome Assembler, 210
- assembly, 112
- association analysis, 377
- attribute, 314
- automated sequencing, 200
- autosome, 7, 368
 - autosomal recessive, 369
- autozygosity, 369
- average, 308
 - genomic recombination rate, 391
 - heterozygosity, 372, 375
 - regional heterozygosity, 372
- B. subtilis*, see *Bacillus subtilis*
- BAC, see bacterial artificial chromo-
some, 418
 - clone, 212
 - library, 212, 217
- Bacillus subtilis*, 2
- background, 302
 - distribution, 249
 - intensity, 304
 - sequence, 254
- bacterial
 - artificial chromosome, 68, 100, 108
 - genome, 216
 - transformation, 60
- bacteriophage, 34
 - lambda, 69
- bactig, 214
- balanced, 130
- barplot, 498
- base
 - calling, 209
 - change, 349
 - composition, 22, 38, 71, 238, 412
 - frequency, 355
 - pair, 21, 23, 99
 - pairing, 32, 299
 - substitution, 353
- basic operation, 105
- Bayes' Theorem, 51, 404
- Bayesian
 - approach, 309, 404
 - computation, 361
- between-genome comparison, 411, 427, 448
- biallelic, 386
- bifurcation, 340
- Big Bang, 425
- big O, 105, 157
- bilateria, 2
- binary
 - search, 171, 172
 - tree, 343
- binding, 225
 - protein, 226
- binomial
 - coefficient, 44
 - distribution, 44, 45, 47, 72, 74, 394, 401
 - random variable, 47
 - success probability, 357
- BioConductor, 319
- biodiversity, 2
- bioinformatics, 333
- biological
 - classification, 337
 - diversity, 263

- replication, 310
- bivalents, 8
- BLAST, 178, 182, 189, 500
- BLASTP, 443
- BLASTX, 437
- block, 186
- BLOCKS, 185
- Blocks database, 185
- blocks substitution matrix, 183, 187
- BLOSUM, *see* blocks substitution matrix
- BLOSUM matrix, 186
- BLOSUM62, 183
- Bonferroni correction, 310
- bottleneck, 387
- bottom-up approach, 110, 195
- Boxplot, 496
- boxplot**, 498
- bp, *see* base pair
- branch
 - length, 340
 - site, 435
- branching topology, 342
- breakpoint, 125, 127
- byrow**, 484

- c, 483
- CAAT box, 435
- Caenorhabditis elegans*, 4
- CAI, *see* codon adaptation index
- CAP, 210
- capillary array electrophoresis, 202
- capture antibody, 330
- carboxy-terminal, 21
- case-control study, 377
- catalytic proteins, 19
- categorical character, 264
- cbind**, 484
- cDNA, 28, 35, 437
 - library, 28
- ceiling**, 492
- cell, 2, 5
 - cycle, 291
 - division, 8
- cellular components, 441
- clusters**, 316
- centimorgan, 9, 379
- Central Limit Theorem, 79
- central tendency, 43

- centroid, 278, 280
- centromere, 418
- challenge set, 233
- character, 13, 263, 264
 - categorical, 264
 - dichotomous, 265, 268
 - quantitative, 265
 - state, 265
- charge-to-mass ratio, 32
- chemical cross-linking, 227
- Chi sequence, 60
- χ^2 test, 89, 360
- child, 343
- ChIP, *see* chromatin immunoprecipitation

- chloroplast, 5, 125
 - genome, 19
- chordata, 2
- chromatid, 8
- chromatin immunoprecipitation, 226
- chromosome, 2, 121
 - rearrangement, 125
 - walking, 219
- cI, 231
- CID, *see* collision-induced dissociation
- cistron, 15
- city-block
 - distance, 270
 - metric, 270
- clade, 340
- cladogenesis, 340
- cladogram, 340
- classification, 253, 254, 263, 286, 313
- clock-like tree, 350, 358
- clone
 - by-clone shotgun, 212, 214
 - coverage, 217
 - library, 109, 216
- cloning, 27
 - vector, 27, 100, 108
- closed architecture, 292, 298
- cluster
 - analysis, 275
 - center, 272
 - membership, 281
 - of orthologous genes, 442, 445
- cluster**, 316
- clustering, 187, 263, 272, 282, 314, 315, 332

- co-regulated, 313
- coalescence, 400, 401
- coalescent, 398, 402
 - tree, 403
- coding, 24
 - region, 361
 - sequence, 15, 432–434
 - strand, 15
- codon, 11, 57
 - adaptation index, 58–60, 62
 - frequencies, 57
 - usage, 24, 25, 57, 58
- coexpressed genes, 324
- COG, *see* cluster of orthologous genes, 444, 445
- coin tossing, 82, 357
 - experiment, 72, 74
- coldspot, 391
- collapsed unitig, 220
- collinear gene cluster, 422, 424, 427
- collision-induced dissociation, 329
- comment, 492
- common ancestor, 354
- comparative genomics, 438
- comparison by content, 170
- complement, 51
- complementary, 20, 299
 - sequence, 32
 - strand, 38
- complete linkage, 273
- compression, 209
- computational problem, 105
- computer code, 105
- conceptual error, 493
- condition, 321
- conditional probability, 51, 53
- confidence interval, 81
- confounded, 360, 398
- consensus, 234, 438
 - sequence, 203, 231, 250, 418
- conserved
 - element, 439
 - linkage, 420
 - segment, 11, 12, 134, 135, 420, 422, 429
 - sequence segment, 424
 - synteny, 9, 11, 123, 135, 420
- contig, 108, 110, 111, 195, 218, 219
- continuous, 265
 - random variable, 76
- control, 308, 330
 - experiment, 303
 - spot, 304
- copy number, 27, 416
- cor, 491
- core
 - promoter, 15
 - proteome, 451
- correctness, 105
- correlation
 - coefficient, 64, 315, 491
 - matrix, 315
- cosmid, 68, 100, 108
 - library, 195
- covariance, 64, 315
 - matrix, 315, 320
- coverage, 109, 114, 168, 203, 208, 217
- CpG island, 416
- criterion, 314
- Cro, 229, 230
- cross
 - linking, 35
 - validation, 237
- crossover, 378
- cutoff, 187, 258, 387
 - score, 253
- Cy3, 299–301, 303, 330
- Cy5, 299–301, 303, 330
- cyanobacteria, 4
- cycle, 130, 320, 343
 - free graph, 343
 - alternating, 133
 - decomposition, 129
- cytoskeleton, 5, 7
 - cytoskeletal proteins, 28
- D*, 381
- D'*, 382
- Danio rerio*, 4
- data
 - mining, 500
 - reduction, 314, 320
 - structure, 242, 314, 321
- database, 167, 499, 501
- DDP, *see* double-digest problem
- debug, 493
- decomposition, 131, 133
 - cycle, 129

- edge-disjoint, 134
- degeneracy, 227
- degree, 343
- degrees of freedom, 309
- deletion, 13, 99, 145
- denaturation, 299
- denature, 30
- dendrogram, 188, 272, 275, 278
- density, 409
- deoxyribonucleotide, 20
- dependent, 42
- detailed balance equations, 356
- detection antibody, 330
- deuterostomes, 2
- `dev.off`, 496
- developmental
 - gene, 299
 - process, 325
 - program, 324
- diagonal, 138, 174
 - sums of scores, 174, 176
- dichotomous character, 265, 268
- dideoxy sequencing, 196, 198, 199
- differential expression, 320
- differentially expressed, 291
- digital sequence tag, 293
- digits, 492
- `dimnames`, 315, 484
- dinucleotide, 24, 48, 416
 - frequency, 49, 242
- diploid, 7
- direct descendant, 358
- directed graph, 343
- discrete, 265
 - distribution, 76
 - random variable, 41
- disease gene, 386
- disjoint, 51
- dissimilarity, 269
 - matrix, 273
- `dist`, 275
- distance, 126, 134, 268, 269, 337, 355, 358
 - matrix, 272, 315, 350, 355
 - method, 346, 350
 - physical, 99
- distinguishability, 269
- distribution, 249
 - binomial, 44, 45, 72, 74, 394, 401
 - discrete, 76
 - exponential, 77, 84, 402
 - geometric, 97
 - hypergeometric, 159
 - normal, 77, 78
 - of restriction sites, 73
 - Poisson, 75, 112, 181, 354, 378
 - posterior, 404
 - uniform, 77
- ditag, 296
- divergence, 143
- divide-and-conquer, 213
- divisive hierarchical methods, 272
- DNA, 21
 - protein complex, 226
 - hybridization, 299
 - microarray, 327
 - polymerase, 196, 197
 - precursor, 197
 - replication, 10, 19
 - sequence assembly, 195
 - sequencing, 34
- domain, 2, 448, 450
 - content, 450
 - structure, 189
- dominant allele, 369
- dot
 - matrix, 173
 - matrix plot, 174
 - plot, 138
- double
 - digest problem, 101, 102
 - crossover, 377
- down-regulated gene, 304
- draft sequence, 219
- Drosophila melanogaster*, 4
- duplicate
 - gene, 427
 - mapping, 430
- duplicated sequence, 121
- duplication, 411, 425
- dye
 - bias, 303
 - fluorescent, 203
 - primer, 203
 - swap, 330
 - terminator, 203
- dynamic
 - programming, 162, 163

- range, 320
- E-value, 181
- E. coli*, *see* *Escherichia coli*
- eBAC, *see* enriched BAC
- EBI, *see* European Bioinformatics Institute
- ecdysozoa, 2
- edge, 129, 219, 343, 443
 - colored graph, 132, 133
 - disjoint, 130
 - alternating cycle, 135
 - cycle, 131
 - decomposition, 134
- edit distance, 265, 273
- efficiency, 105
- electrophoresis, 30
- electrophoretic mobility-shift assay, 226
- electrostatic interaction, 227
- emission maximum, 300
- empty set, 349
- enriched BAC, 214
- Ensemble, 503
- Entrez, 500
- enzyme, 19
- epitope, 33, 330
- error, 277
 - message, 493
 - Type I, 252
 - Type II, 252
- Escherichia coli*, 2
- EST, *see* expressed sequence tag, 293, 437
- eubacteria, 2
- euchromatin, 219, 418
- Euclidean distance, 269, 270, 280
- eukaryote, 2, 5, 451
 - eukaryotic
 - cells, 5
 - genes, 15
 - promoter sequence, 435
- EULER, 210
- European Bioinformatics Institute, 500
- evolution, 9, 17, 367, 418
- evolutionary
 - clock, 354
 - distance, 188, 267, 442
 - process, 353
 - relationship, 143, 367
 - tree, 342
- exact occurrences, 96
- exhaustive, 51
- exon, 15, 435
 - and intron statistics, 437
 - size distribution, 437
- ExpASy, 503
- expectation, *see* expected value
- expected
 - heterozygosity, 396
 - number of islands, 114
 - number of occurrences, 89
 - value, 43
- experimental design, 310, 312
- exponential
 - distribution, 77, 84, 402
 - growth, 387
 - random variable, 77
- expressed sequence tag, 28
- expression
 - matrix, 321
 - pattern, 313, 324, 325, 327, 332
 - profile, 325
 - ratio, 308, 310, 321
 - vector, 30
- extinction, 18
- F-statistic, 376
- failure, 81
- false
 - discovery rate, 254
 - negative, 252, 253
 - assignment, 253
 - positive, 189, 252, 253, 309, 320
 - assignment, 253
- FASTA, 173, 178, 182, 189, 500
- FDR, *see* false discovery rate
- feature, 299, 309
- Felsenstein's model, 354
- fingerprint, 68, 115, 214
- finished sequence, 219
- first-order Markov chain, 52
- fitness, 384
- fixation, 394, 396
 - index, 376
- floor, 492
- fluorescence, 320
 - intensity, 300
- fluorescent dye, 203, 299, 330

- fold, 24
- footprint, 226
- footprinting, 35, 226
- fossil record, 337
- founder effect, 384
- four-point condition, 350, 352
- fr(G + C), 38
- fractional overlap, 114
- fragment
 - length distribution, 87
 - order, 103
- frameshift mutation, 161
- function, 168, 441, 443, 495
- functional
 - annotation, 168, 442
 - categories, 445
 - domain, 445
 - element, 439
 - proteomics, 327
- fungi, 2

- gamete, 8, 9
- gap, 114, 161, 361
 - closure, 110
 - length penalty, 176
 - penalty, 161
- GC**
 - box, 435
 - content, 412
 - skew, 39, 61, 412
- gel electrophoresis, 30, 31, 35, 68, 101
- gel-shift, 35
 - assay, 226
- gene, 13, 368
 - specific probe, 299
 - chip, 299
 - experiment, 35
 - content, 448
 - copy number, 30
 - down-regulated, 304
 - expression, 291, 324
 - analysis, 332
 - data, 322
 - matrix, 303, 314
 - pattern, 325
 - fusion, 444
 - homeotic, 168
 - neighbor, 444
 - ontology, 449
 - orientation, 132
 - pool, 393
 - prediction tools, 436
 - recognition, 25
 - regulation, 35, 313
 - tree, 361, 362
 - up-regulated, 304
- Gene Ontology Consortium, 441, 448
- genealogical history, 385
- general hypothesis, 360
- generation, 362
- genetic
 - code, 4, 24, 397, 508
 - cross, 9
 - disease, 386
 - diversity, 342
 - drift, 384, 393–395
 - exchange, 378
 - linkage, 11
 - map, 9, 68, 121, 386
 - distance, 378
 - mapping, 34, 419
 - marker, 9, 99, 379
 - network, 324
 - variation, 18, 369
- genome, 1, 19, 121, 414
 - chloroplast, 19
 - equivalent, 109
 - human, 13
 - mammalian, 28
 - metazoan, 440
 - mitochondrial, 19
 - nuclear, 19
 - rearrangement, 123, 135, 425
 - sequence assembly, 219
 - sequencing, 213
 - signature, 452
 - size, 108, 412, 440
- genomic library, 108
- genotype, 18, 377
- geographic regions, 370
- geometric
 - distribution, 97
- geometric mean, 59
- germline, 8
- getwd**, 486
- global
 - alignment, 145, 147, 152, 163
 - number of, 157

- normalization, 304, 514
- Golgi apparatus, 5
- graph, 129, 343
 - edge-colored, 132, 133
- graphics.off**, 496
- greedy procedure, 205
- group average, 273
 - linkage, 351
- guilt by association, 313

- Haldane mapping function, 378
- half
 - bits, 187
 - site, 229
- Hamiltonian path problem, 215, 216
- Hamming distance, 337, 353, 357
- haploid, 7
- haplotype, 9, 11, 376, 377, 385
 - block, 387, 388, 390
 - frequency, 387
- Hardy-Weinberg law, 396
- hclust**, 276, 322
- help**, 56, 481
- help.start**, 481
- heterochromatin, 219, 418
- heterozygosity, 371, 372, 375, 398
- heterozygote, 369
- heuristic, 162
 - search, 349, 359
- hidden Markov model, 162, 252, 504
- hierarchical
 - classification, 263
 - cluster, 337
 - clustering, 272, 315, 322, 350, 353
- high-copy-number plasmid library, 217
- high-scoring
 - diagonal, 177
 - segment pair, 179, 189
- highest-scoring
 - alignment, 146
 - path, 153
- hist**, 47, 81, 241, 497
- histogram, 45, 496, 497
- histones, 6, 28
- hit, 180, 443
- HIV, 339
- HIV-I, 338
- homeotic gene, 168
- homodimer, 229

- homogeneous, 52
 - chain, 242
- homolog, 143, 168, 189, 442
- homologous recombination, 8, 13
- homology, 143
- homoplasmy, 143
- homopolymer, 39
- homozygosity, 375
- homozygous, 369
- horizontal transfer, 414
- host
 - organism, 27
 - specificity, 27
- housekeeping gene, 299
- HSP, *see* high-scoring segment pairs
- human
 - genome, 13
 - immunodeficiency virus, 338
 - populations, 338, 340, 369, 386
- Human Genome Project, 68, 212
- hybrid strategy, 214
- hybridization, 299, 381
 - efficiency, 303
- hybridoma, 33
- hydrodynamic shear, 67
- hydrogen
 - bond
 - donor, 23, 228
 - receptor, 23
 - bonding, 227
- hydrophobic interactions, 227
- hypergeometric distribution, 159
- hypothesis
 - general, 360
 - reduced, 360
 - test, 308
- hypothetical protein, 452

- identity, 146, 149, 152
 - permutation, 126
- iid, *see* independent and identically distributed
 - model, 72, 238
- immune
 - response, 32
 - system, 32
- immunoglobulin, 32
- in-frame fusion, 328
- incomplete

- digestion, 106, 110
- dominance, 369
- incorporation efficiency, 303
- indel, 14, 145, 146, 149, 153
- independence, 42
- independent
 - and identically distributed, 42
 - parameters, 237
- indirect labeling, 330
- infinitely many
 - alleles model, 397
 - sites model, 401
- information, 249, 250
 - content, 236
 - theory, 248
- inheritance, 7
- initial
 - distribution, 55
 - probability distribution, 53, 244
- input, 105
- insects, 2
- insertion, 13, 99, 145
- intensity
 - dependent normalization, 304–306
 - ratio, 309, 310, 312
- inter-cluster distance, 277
- intercalation, 320
- interleaved position, 103
- interleaving of genes, 429
- internal
 - branch, 343
 - node, 343, 358
- Internet, 499
- interphase, 8
- InterPro, 504
- intersect, 349
- intersection, 51, 349
- interval, 155
- intron, 15, 435
- inverse, 490
- inversion, 13, 99, 123, 125, 145
- inverted repetition, 73, 227
- `is.data.frame`, 486
- `is.matrix`, 490
- island, 111, 112
- isoelectric
 - focusing, 30, 31
 - point, 30, 31
- IUPAC-IUB symbols, 507
- Jaccard's coefficient, 269
- jackknife procedure, 278
- joint probability distribution, 64
- Jukes-Cantor formula, 357
- K*-means, 272, 278, 282, 318
 - clustering, 279
- k*-tuple, 38, 169, 173, 251
 - distribution, 60
- k*-word, 38, 89, 94, 411
 - composition, 412
 - frequency, 92
- kb, *see* kilobase pairs
- keyword, 447, 500
- kilobase pairs, 20, 99
- kilodalton, 33
- kinetic PCR, 320
- `kmeans`, 283, 316
- Kullback-Leibler distance, 249
- lagging strand, 60, 412
 - DNA synthesis, 412
- lambda vector, 100
- landmark, 11, 135
 - sequence, 12
- law of total probability, 51, 53
- layout, 203, 204
- LC/MS, 328
- LD, 381, *see* linkage disequilibrium,
 - 384, 391, 405
 - decay, 383
- leading strand, 39, 60
 - DNA synthesis, 412
- leaf, 343, 358
- learning, 233
- leaves, 340
- length**, 85, 482, 493, 495
- letter occurrence, 96
- Levenshtein distance, 265
- library, 27
 - screening, 117
- library**, 495
- likelihood
 - based method, 346
 - function, 83
 - methods, 361
- LINE, 417, 419
- linear model, 491

- lines, 81, 497
- linkage, 376
 - analysis, 376
 - disequilibrium, 381, 385, 388
 - equilibrium, 382
- linker, 296
- liquid chromatography, 328
- list, 92, 170, 239
 - ordered, 172
- list, 315, 484
- lm, 491
- local
 - alignment, 145, 155, 156, 163
 - pairwise, 163
 - score, 181
 - match, 179
 - scope, 493
- locally weighted scatterplot smoother, 305
- locus, 13, 368
- loess, 305, 306
- log-likelihood, 360
 - function, 83
- logical comparison, 488
- look-up table, 176
- lophotrochozoa, 2
- low-copy-number plasmid library, 217
- lowess, 305
- LTR transposon, 417, 419
- lty, 497

- M. genitalium*, see *Mycoplasma genitalium*
- MA plot, 304, 306
- macromolecular complexes, 30
- macrorearrangement, 136
- major groove, 21, 23, 227
- MALDI, 328, see matrix-assisted laser-induced desorption/ionization
 - MS, 328
 - TOF, 381
- mammal, 2
 - mammalian
 - cell, 27
 - genome, 28
- Manhattan distance, 270
- map
 - based, 195
 - distance, 379
 - genetic, 68, 121
 - optical, 100
 - physical, 67–69, 99, 121
 - restriction, 67, 99
- marker, 99
 - genetic, 99
- Markov
 - chain, 50, 242–244, 396
 - first-order, 52
 - Monte Carlo, 361
 - second-order, 59
 - model, 54, 244
 - property, 52, 53
- mass spectrometry, 35, 328
- match, 146, 149
 - model, 182
- mate pair, 217, 218
- maternally inherited, 338
- matlines, 498
- matplotlib, 498
- matrix
 - alignment, 204
 - computation, 489
 - overlap, 205
 - scoring, 184
 - substitution, 183
- matrix, 56, 90, 94, 245, 394, 484
- matrix-assisted laser-induced desorption/ionization, 328
- max, 153, 156
- max, 87
- maximum likelihood, 83, 358
 - estimator, 83, 359
- Mb, see megabase pairs
- mean, 43, 491
 - number of substitutions, 355
- mean, 305, 487, 492
- MEGA, 351
- megabase pairs, 20
- meiosis, 6–10, 369, 377, 379
- Mendelian genetics, 368
- metazoan genome, 440
- methylation, 69
- metric, 269
- mfcol, 498
- mfrow, 138, 318, 498
- microarray, 27, 298
 - data, 512
- microrearrangement, 136

- microsatellite, 370, 380
 - repeat, 386
 - repeats, 161
- min, 158
- min**, 87
- minimal
 - genome, 440
 - tiling
 - clone set, 115
 - path, 111
 - set, 214
- minisatellite, 380
- Minkowski metric, 271
- minor
 - allele, 374
 - groove, 21, 23, 227
- mismatch, 146, 149, 152
 - penalty, 160
 - sequence, 180
- mismatched base, 381
- mitochondria, 5, 100, 125
 - mitochondrial DNA, 27, 338
 - mitochondrial genome, 19
- mitosis, 6–8
 - mitotic spindle, 7
- mixing of populations, 384
- MLE, *see* maximum likelihood estimator
- model
 - for DNA sequence, 71
 - match, 182
 - organisms, 4, 167
 - random, 182
- molecular
 - clock, 350
 - cloning, 67
 - evolution, 18
 - function, 441, 448
 - mass, 30
- monoclonal antibody, 33
- most recent common ancestor, 340
- motif, 448
- mRNA, 5, 13, 19, 28, 298, 332
 - abundance, 292, 296
 - processing, 292
- MS, *see* mass spectrometry
- mtDNA, *see* mitochondrial DNA, 338, 340
- m/z , 329
- multiple
 - sequence alignment, 145
 - alignment, 161, 203, 205, 438
 - hypothesis testing, 309
- multiplication rule, 42
- multivariate, 264
 - data, 321
- Mus musculus*, 4
- mutation, 11, 145, 348
 - mutational load, 418
 - parameter, 401
 - rate, 386, 402
- Mycoplasma genitalium*, 49, 55
- $N(0, 1)$, *see* standard normal distribution
- $N(\mu, \sigma^2)$, *see* normal distribution
- National Center for Biotechnology Information, 499
- natural selection, 18, 368, 384
- NCBI, *see* National Center for Biotechnology Information
- nclass**, 498
- ncol**, 484
- neighbor joining, 351
- neighborhood
 - sequence, 179
 - size, 180
- Neisseria gonorrhoeae*, 60
- network, 332, 444
- neutral mutation, 397
- node, 348
- non-hierarchical clustering, 278
- noncoding
 - region, 417
 - sequence, 432, 433
- nonsite, 252
- nonspecific binding, 302
- normal distribution, 77, 78
- normalization, 303, 515
 - intensity-dependent, 304–306
 - global, 304
- normalizing constant, 404
- NP-complete, 106, 216
- nuclear genome, 19
- nuclei, 2
- nucleoid, 5
- nucleosome, 20
- nucleotide, 20

- diversity, 401
- null
 - hypothesis, 252, 308
 - model, 92
 - set, 89
- number of
 - alignments, 159
 - branches, 345
 - clusters, 276
 - differences, 400
 - genes, 440
 - global alignments, 157
 - islands, 112
 - mutations, 400
 - parameters, 242
 - rooted trees, 346
 - trees, 344
 - unrooted trees, 346
- object, 263, 264, 481
- objects**, 481
- Ockham's Razor, 347
- odds ratio, 234
- offset, 174, 178
 - diagonal, 178
- oligomer, 227
- oligonucleotide
 - array, 381
 - chip, 293
 - probe, 381
- one-
 - step transition matrix, 52
 - tailed p-value, 95
- open
 - architecture, 292, 327
 - reading frame, 25, 26
- operational taxonomic unit, 264
- operator, 17, 231
- operon, 17, 444
- optical
 - detection, 298
 - mapping, 100, 106
- optimization, 272, 278
- options**, 492
- ordered list, 172
- ORF, *see* open reading frame
- organelles, 2
- organisms, 1
- ortholog, 144, 442, 444
 - orthology, 441, 443
- OTU, *see* operational taxonomic unit, 337
- Out-of-Africa, 340
- outcomes, 72
- outgroup, 342
- outlier, 279
- output, 105
- overcollapsed unitig, 219
- overlap, 108, 111, 112, 115, 204, 217
 - alignment, 203
 - matrix, 205, 207
- p-value, 82, 95, 180
- paired-end sequence, 217
- pairwise
 - comparison, 203, 204
 - linkage disequilibrium, 389
 - local alignment, 163
- palindrome, 73
- PAM, *see* point accepted mutation
- par**, 138, 305, 318, 496
- paralog, 144, 425, 442
- paralogous gene pair, 429
- parameter, 75
 - estimation, 83, 360
 - number of, 242
- parent, 343
- parsimonious alignment, 145
- parsimony, 342, 346, 348
 - cost, 349
- partial restriction map, 117
- partition, 51, 280
- parts inventory, 332
- path, 157
 - highest-scoring, 153
- pattern
 - discovery, 96, 229
 - recognition, 34
- PCA, *see* principal components analysis
- pch**, 138, 496
- PCR, *see* polymerase chain reaction, 294, 320, 380
- Pearson product-moment correlation
 - coefficient, 315, 382
- pedigree, 376
- peeling algorithm, 359
- penetrance, 369
- permutation, 100, 121, 125, 134

- identity, 126
 - worst-case, 129
- perspective plot, 496
- phage, 34
 - display, 34, 330
- phagemid, 196
- phenotype, 13, 368, 380
- phosphodiester backbone, 21, 23, 227
- photolithography, 299
- photosynthesis, 4
- Phrap, 210
- Phred, 209
- phylogenetic
 - footprinting, 439
 - profile, 444, 446
 - relationship, 360
 - tree, 18, 34, 126, 278, 339, 340, 367
- phylogeny, 126, 337, 360
- physical
 - distance, 99
 - map, 67–69, 99, 121, 420
 - mapping, 34
- pI, *see* isoelectric point
- piechart, 496
- pin, 496
- plant, 2
- plasma membrane, 5
- plclust, 276, 322
- plot, 138, 258, 305, 318, 496
- pnorm, 78, 95
- point, 368
 - accepted mutation, 183
 - mutation, 13
- points, 258, 318, 497
- Poisson
 - approximation, 74, 75, 401
 - distribution, 75, 112, 181, 354, 378
 - process, 76, 84, 403
- poly-A, 299
 - tail, 17
- polyacrylamide gel, 198
- polyadenylation, 17
- polycistronic, 15
- polyclonal antibody, 32
- polygenic traits, 369
- polymerase chain reaction, 27, 380
- polymorphic
 - locus, 384
 - marker, 379
- polymorphism, 380
- polypeptide, 21, 328
- pooled population, 376
- population, 18, 159, 337, 362, 367, 370, 400
 - expansion, 405
 - genetics, 18, 367
 - history, 368
 - mean, 308
 - model, 384
 - size, 386, 399
 - stratification, 385
 - structure, 374, 384
 - variance, 308
- position-specific scoring matrix, 235
- positional weight matrix, 233, 236, 238, 242, 251, 254, 257, 435, 436
- posterior
 - distribution, 404
- ppois, 75
- pre-finished sequence, 219
- predict, 307
- predictor variable, 314
- primary structure, 24
- primer, 196
 - extension, 381
 - universal sequencing, 197
- principal components analysis, 314, 319
- prior probability distribution, 309
- probability, 40
 - density function, 76, 77, 400
 - distribution, 41, 44, 232
 - background, 238
 - initial, 244
 - joint, 64
 - prior, 309
 - mass function, 41
- probe, 298
- processes, 441
- profile, 251, 314, 436, 448
- program, 495
- prokaryote, 2, 5, 7
 - prokaryotic
 - chromosome, 7, 412
 - gene, 15
- promoter, 15, 89, 225, 234, 235, 435, 509
- prophase I, 8
- PROSITE, 185

- protein, 21
 - DNA complex, 35
 - protein interaction, 35, 327, 329
 - alignment, 184
 - binding, 226
 - expression, 327
 - microarray, 329
 - modification, 292
 - regulatory, 225
- proteome, 35, 291, 327, 450
- protists, 2
 - Protista, 2
- protostomes, 2
- pseudo-random number, 45
- pseudocount, 237
- PSSM, *see* position-specific scoring matrix
- punctate recombination, 387, 391
- PWM, *see* positional weight matrix

- q, 480
- qualitative, 264
- quality
 - file, 211
 - score, 209
- quantitative character, 265
- quaternary structure, 24
- query, 167
 - sequence, 179

- R, 479
 - prompt, 480
- random
 - assortment, 379
 - bifurcating tree, 402
 - model, 182
 - number generator, 46
 - rate variation, 361
 - variable, 41, 76
- rare
 - allele, 387
 - word, 171
- ratio of fluorescence intensities, 303
- rbind**, 484
- rbinom**, 47, 80
- read depth, 214, 219, 422, 425
- read.table**, 244, 282, 485, 487, 491, 513
- real-time
 - PCR, 320
 - RT-PCR, 320
- rearrangement, 139, 420
- reassigning objects, 277
- recessive allele, 369
- reciprocal best hit, 424
- recognition
 - sequence, 70, 71
 - sites, 60
- recombination, 7–9, 60, 376, 377, 405
 - coldspot, 391
 - fraction, 378, 383, 386
 - frequency, 379
 - homologous recombination, 8, 13
 - hotspot, 391
 - parameter, 383, 386
 - rate, 379, 402, 405
 - variation, 391
- recursion, 162
 - formula, 383
- reduced hypothesis, 360
- reductionist approach, 332
- reference sample, 312
- region, 370
- regulatory
 - protein, 225
 - sequence, 251
- rejection algorithm, 405
- relative entropy, 249
- rep**, 86, 394, 494
- repeated sequence, 215, 216, 416
- replicate, 307, 310
- replication, 4
 - biological, 310
 - origin, 39, 225
 - slippage, 161
 - technical, 310
- resolution, 392
- respiration, 5
- response variable, 314
- restriction
 - endonuclease, 34, 67, 68, 70, 71, 99
 - Type II, 70
 - enzyme, 69, 99
 - fragment length
 - distribution, 84
 - polymorphism, 380
 - map, 67, 99
 - mapping, 34, 107

- Smith-Birnstiel, 200
 - site, 71
 - site polymorphism, 99
- retrovirus, 338
- return**, 493, 495
- reversal, 100, 102, 123, 126, 127, 420
 - distance, 132, 134
- reverse
 - complement, 204, 205
 - transcriptase, 197
 - transcription, 19, 28
- reversible, 356
- rexp**, 409
- RFLP, *see* restriction fragment length polymorphism
- rm**, 487
- rnorm**, 498
- robust, 277, 286
- root, 343
 - node, 343
- rooted tree, 343
- round**, 492
- rpois**, 409
- rRNA, 19
- r^2 , 383
- runif**, 409

- Saccharomyces cerevisiae*, 4
- SAGE, *see* serial analysis of gene expression
- sample
 - correlation, 65
 - covariance, 65
 - size, 387
 - small, 237
 - space, 51
 - variance, 45
- sample**, 46, 56, 85, 94, 255
- sampling, 159
 - variation, 384
- scaffold, 214, 218, 219
- scale, 282
- scaling, 271
- scan**, 315, 482, 486, 487
- scatterplot, 305, 496
- Schizosaccharomyces pombe*, 4
- score, 233, 234
 - alignment, 153
 - distribution, 253, 255, 257
- scoring
 - matrix, 147, 160, 176, 181, 184
 - rules, 160
 - system, 146
- search space, 167
- second-order Markov chain, 59
- secondary structure, 21, 22, 24, 361
- segmental duplication, 13, 145, 420, 422, 423, 425
- segregating site, 401, 403
- selection, 189, 266
- sense strand, 26
- sensitivity, 189, 252
- sequence
 - tagged
 - connector, 117, 212, 213
 - site, 99
 - anchor, 135
 - assembler, 210
 - assembly, 207, 211
 - coverage, 212, 214, 219
 - divergence, 418
 - file, 210, 211
 - identity, 422
 - logo, 250
 - mismatch, 180
 - neighborhood, 179
 - paired-end, 217
 - pattern, 225
 - query, 179
 - read, 208
 - similarity, 174, 443, 500
 - search, 437
 - trace, 209
 - variation, 338
- sequence-tagged
 - connector, 212
 - site, 368
- sequencing
 - automated, 200
 - dideoxy, 196
 - strategy, 212
 - technology, 195
- serial analysis of gene expression, 35, 293, 297, 327
- set.seed**, 46
- setwd**, 486
- sex chromosomes, 7
- Shannon's entropy, 248

- shotgun sequencing, 68, 168, 196, 203, 212
- signal, 96, 225, 232, 249
- signif**, 241
- significance level, 310
- significant diagonal, 177
- simian immunodeficiency virus, 338, 339
- similarity, 143, 144, 268
 - coefficient, 265
 - matrix, 269
- simple matching coefficient, 268
- simulate, 236, 254, 427
- simulated sequence, 40
- simulation, 40, 45, 47, 85, 93, 94, 136, 139, 394
 - algorithm, 40
- SINE, 417, 419
- single
 - linkage clustering, 273
 - nucleotide polymorphism, 12, 381, 386
 - crossover, 377
 - linkage, 273
- singleton, 111, 112, 114
- sink**, 487
- sister chromatid, 8
- site, 252
 - distribution, 71
- SIV, *see* simian immunodeficiency virus
- size**, 316
- size selection, 110
- skip**, 486
- slab gel, 202
- slippage, 209
- Slow Shuffle, 425
- small sample correction, 245, 246
- Smith-Birnstiel restriction mapping, 200
- Smith-Waterman, 189
 - alignment, 504
 - local alignment, 163, 173
- SNP, *see* single nucleotide polymorphism
- solve**, 489
- somatic cells, 7, 8
- source**, 492
- space efficiency, 104, 106
- speciation, 18, 368
- species
 - tree, 361, 362
- specificity, 189, 252
- splice junction, 435
- splicing, 15, 435
- spot intensity, 304
- spotted microarray, 35, 293, 301
- spread, 43
- stable, 278
- standard
 - deviation, 44, 73, 271, 307, 491
 - normal distribution, 77, 78, 80, 82, 94
- standardize, 80, 316
- stationary, 355
 - base frequencies, 359
 - distribution, 54
- statistic, 308
- statistical significance, 94
- STC, *see* sequence-tagged connector
- steady-state, 397
- Stirling's approximation, 158
- stochastic
 - model, 353, 392
 - process, 353
- stop codon, 24
- stratification, 384
- stratified, 374
- string, 70, 144, 155
- structural proteins, 19
- STS, *see* sequence-tagged site
- Student's *t* statistic, 308
- subpopulation, 368, 374, 375, 385
- subsequence library, 298
- substitution, 145, 146, 149
 - matrix, 183
- subtree, 359
- success, 72, 81
- sum**, 91, 493
- supervised, 313
 - learning, 231
- surprise, 82, 181
- SWISS-PROT, 184
- Swiss-Prot, 503
- symmetry, 269
- synonymous change, 266
- syntactical error, 493
- syntenic, 420
- synteny
 - block, 137

- syntenic block, 11, 121, 122, 136
 - syntenic segment, 11, 12
- systematics, 448
- systems biology, 332
- tag SNP, 386
- tagging
 - enzyme, 294, 296
 - restriction endonuclease, 294
- tandem
 - mass spectrometry, 329
 - repeat, 370
- target, 145, 167, 298, 329
- TATA**
 - less promoter, 435
 - box, 435
- taxa, 263, 337, 367
- taxon, 263, 367
- technical replication, 310
- telomere, 123
 - telomeric repeat, 416
- template, 196
 - strand, 15, 26
- termination site, 435
- tertiary structure, 22, 24
- text editor, 244
- The Institute for Genomic Research, 503
- third position, 361
- three-point condition, 350
- threshold, 170, 177, 180
 - cycle, 320
- TIGR, *see* The Institute for Genomic Research
- TIGR assembler, 210
- time, 353
 - course, 318
 - efficiency, 104
- time to most recent common ancestor, 398
- tissue culture, 6
- title**, 318, 496
- T_m , 299
- T_{MRCA} , 398, 399, 401, 402
- TOGA, *see* total gene expression analysis
- top-down, 195, 212
- total gene expression analysis, 35, 293, 295
- t' , 308
- tracing back, 151
- training set, 233, 237, 252, 254
- transcript, 327
 - abundance, 292
- transcription, 19
 - factor, 251, 324
 - binding site, 252
 - reverse, 19
 - transcriptional start site, 435
- transcriptome, 35, 292, 327
- TRANSFAC database, 237
- transition, 161
 - matrix, 52, 54, 55, 242, 243, 246, 396
- translation, 5, 19
 - translational terminators, 15
- translocation, 13, 121, 125, 127, 145
- transposable element, 13, 380, 411, 416, 417, 419
- transversion, 161
- traveling salesman problem, 105
- treatment, 308, 330
- tree, 2, 162, 343, 350
 - building, 361
 - construction, 342
 - gene, 361, 362
 - species, 361, 362
 - topology, 359
- triangle inequality, 269
- trinucleotide, 24
- triplet, 24
- tRNA, 19
- true positive, 189, 309
- TSP, *see* traveling salesman problem
- twofold rotational symmetry, 229
- type**, 496
- Type I error, 252
- Type II
 - error, 252
 - restriction endonuclease, 70
- Type IIS endonuclease, 296
- U-unitig, 219
- ultrametric
 - condition, 352
 - tree, 350, 351
- unbiased, 404
- undebug**, 493
- ungapped extension, 180

- uniform
 - distribution, 77
 - random number, 46
- uniformly distributed, 45
- union, 51, 349
- unique sequence, 215, 219, 416
- unitig, 217, 218
- universal sequencing primer, 197
- unlinked, 379
- unrooted, 344
- unsupervised, 314
 - approach, 229
- unweighted pair group method with
 - arithmetic means, 351
- up-regulated gene, 304
- UPGMA, *see* unweighted pair group
 - method with arithmetic means
- uracil, 21

- validation set, 233, 237
- Var, *see* variance
- var**, 316
- variable number of tandem repeats, 99
- variable number tandem repeats, 380
- variance, 43, 45, 73, 310, 312, 491
- variation, 9
- vector component, 321
- vertex, 129, 219, 343
- VNTR, *see* variable number of tandem
 - repeats

- Watterson estimator, 404
- weighted
 - least squares, 351
 - parsimony, 350
- WGS, *see* whole-genome shotgun
- while**, 494
- whole-genome
 - duplication, 425, 428
 - shotgun, 196, 212, 213
 - assembly, 422
 - sequence assembly, 168
 - sequencing, 214, 425
- Whole-genome Assembler, 210
- window, 233
- within
 - cluster sum of squares, 285, 317
 - genome
 - analysis, 448
 - comparison, 411, 420
 - ss, 279, 285
- withinss**, 316
- word, 38
 - content, 169
 - count, 91, 96
 - decomposition, 96
 - frequency, 38
 - list, 170, 172
 - overlap, 93, 96
 - rare, 171
- workspace, 486
- worst-case permutation, 129
- Wright-Fisher model, 392, 393, 397
- write.table**, 487

- X^2 , 49
- xlab**, 496
- xlim**, 496

- YAC, *see* yeast artificial chromosome
- YAC library, 195
- yeast artificial chromosome, 68, 108
- Yersinia*, 100, 414
 - pestis*, 414
- ylab**, 496
- ylim**, 496
- Y. pestis*, *see Yersinia pestis*

- Zea mays*, 4
- zygote, 8