Bernhard Haubold
Thomas Wiehe

Introduction to Computational Biology

An Evolutionary Approach

Birkhäuser

Bernhard Haubold
Thomas Wiehe

Introduction to Computational Biology

An Evolutionary Approach

Birkhäuser Verlag
Basel · Boston · Berlin

Bernhard Haubold
Department of Biotechnology and
Informatics
University of Applied Sciences
Weihenstephan
85350 Freising
Germany

Thomas Wiehe
Institut für Genetik
Universität zu Köln
Zülpicher Strasse 47
50674 Köln
Germany

A CIP catalogue record for this book is available from the Library of Congress, Washington D.C., USA

Bibliographic information published by Die Deutsche Bibliothek
Die Deutsche Bibliothek lists this publication in the Deutsche Nationalbibliografie;
detailed bibliographic data is available in the Internet at <http://dnb.ddb.de>.

ISBN 10: 3-7643- 6700-8
ISBN 13: 978-3-7643-6700-8
Birkhäuser Verlag, Basel – Boston – Berlin

The publisher and editor can give no guarantee for the information on drug dosage and administration contained in this publication. The respective user must check its accuracy by consulting other sources of reference in each individual case.
The use of registered names, trademarks etc. in this publication, even if not identified as such, does not imply that they are exempt from the relevant protective laws and regulations or free for general use.
This work is subject to copyright. All rights are reserved, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, re-use of illustrations, recitation, broadcasting, reproduction on microfilms or in other ways, and storage in data banks. For any kind of use, permission of the copyright owner must be obtained.

© 2006 Birkhäuser Verlag, P.O. Box 133, CH-4010 Basel, Switzerland
Part of Springer Science+Business Media
Printed on acid-free paper produced from chlorine-free pulp. TCF ∞
Printed in Germany
Cover illustration: Simulation of gene genealogies under the Wright-Fisher model of evolution. Each gene (dot) is linked to exactly one ancestral gene in the proceding generation. In addition, it may be linked to one or more descendants in subsequent generations.
ISBN 10: 3-7643-6700-8 e-ISBN 10: 3-7643-7387-3
ISBN 13: 978-3-7643-6700-8 e-ISBN 13: 978-3-7643-7387-0

9 8 7 6 5 4 3 2 1 www.birkhauser.ch

To Angelika and Claudia

Preface

In 1982, the first release of the GenBank sequence database contained 601,438 residues. By 2005, this number had grown beyond 10^{11} and continues to increase exponentially. Far from regarding this as "information overload", we believe the free availability of so much precise and fundamental data on the ultimate constituents of life to be the hallmark of a golden age in biomedical research. Computational biology is concerned with helping to understand these data.

The aim of this book is to give a first introduction to the computational aspects of genome-scale molecular biology, also known as genomics. The interpretation of biological data is often contingent on an understanding of the evolutionary history that has generated it. Hence, we explain evolutionary models as well as classical sequence analysis.

Our intended audience is primarily students of bioinformatics, as well as researchers and students in neighboring disciplines including molecular biology, genetics, medicine, physics, mathematics, and computer science. As background, we assume familiarity with basic general and molecular biology as well as elementary probability theory. We also expect an interest in computers and their programming.

In writing this book we have benefited from the expertise and support of a number of colleagues. Clemens Beckstein invited us in 1999 to give our first lecture series on computational biology at Jena University. Wolfgang Stephan and Monty Slatkin encouraged us to turn the lecture notes accumulated in Jena into a textbook. Steffi Gebauer-Jung helped with some of the algorithms we present. Claudia Acquisti, Frank Leßke, Peter Pfaffelhuber, Karl Schmid, and Daniel Zivkovic commented on earlier versions of the manuscript. Our students improved our teaching of computational biology over the years. Finally, we owe a huge debt of gratitude to Angelika Börsch-Haubold, who edited the entire book, compiled the index and guided this project through the production stage. Without her contribution there would be no book.

Freising and Köln,
March 2006

Bernhard Haubold
Thomas Wiehe

Contents

1 Introduction 1
  1.1 Reading and Writing 1
  1.2 Design and Scope of This Book 3
    1.2.1 Sequences in Space 4
    1.2.2 Sequences in Time 7

Part I Sequences in Space

2 Optimal Pairwise Alignment 11
  2.1 What Is an Alignment? 14
  2.2 Biological Interpretation of the Alignment Problem 15
  2.3 Scoring Alignments 15
  2.4 Amino Acid Substitution Matrices 16
    2.4.1 PAM Matrices 18
    2.4.2 BLOSUM Matrices 22
    2.4.3 Comparison between PAM and BLOSUM 25
    2.4.4 Application of Substitution Matrices 27
  2.5 The Number of Possible Alignments 27
  2.6 Global Alignment 30
  2.7 Shotgun Sequencing and Overlap Alignment 33
  2.8 Local Alignment 35
  2.9 Accommodating Affine Gap Costs 36
  2.10 Maximizing *vs.* Minimizing Scores 38
  2.11 Example Application of Global, Local, and Overlap Alignment . . . 39
  2.12 Summary 39
  2.13 Further Reading 40
  2.14 Exercises and Software Demonstrations 40

3 Biological Sequences and the Exact String Matching Problem 43
- 3.1 Exact *vs.* Inexact String Matching 43
- 3.2 Naïve Pattern Matching 44
- 3.3 String Searching in Linear Time 45
- 3.4 Trees 46
- 3.5 Set Matching Using Keyword Trees 48
- 3.6 Suffix Trees 51
- 3.7 Suffix Tree Construction 54
- 3.8 Suffix Arrays 55
- 3.9 Repetitive Sequences in Genomics—the C-value Paradox 56
- 3.10 Detection of Repeated and Unique Substrings Using Suffix Trees . 57
- 3.11 Maximal Repeats 59
- 3.12 Generalized Suffix Tree 59
- 3.13 Longest Common Substring Problem 60
- 3.14 k-Mismatches 60
- 3.15 Summary 62
- 3.16 Further Reading 62
- 3.17 Exercises and Software Demonstrations 63

4 Fast Alignment: Genome Comparison and Database Searching 65
- 4.1 Global Alignment 67
- 4.2 Local Alignment 69
  - 4.2.1 Global/Local Alignment: k-Error Matching 71
  - 4.2.2 Examples of Database Search Programs 73
- 4.3 Database Composition 79
- 4.4 Heuristic *vs.* Optimal Alignment Methods 79
- 4.5 Application: Determining Gene Families 79
- 4.6 Statistics of Local Alignments 81
  - 4.6.1 Maximum Local Alignment Scores 81
  - 4.6.2 Choosing a Substitution Matrix 84
- 4.7 Bit Scores 85
- 4.8 Summary 85
- 4.9 Further Reading 86
- 4.10 Exercises and Software Demonstrations 86

5 Multiple Sequence Alignment 91
- 5.1 Scoring Multiple Alignments 94
- 5.2 Multiple Alignment by Dynamic Programming 94
- 5.3 Heuristic Multiple Alignment 97
- 5.4 Summary 98
- 5.5 Further Reading 99
- 5.6 Exercises and Study Questions 99

6 Sequence Profiles and Hidden Markov Models 101
6.1 Profile Analysis 101
6.2 Hidden Markov Models 106
6.3 Profile Hidden Markov Models 111
6.4 Summary 113
6.5 Further Reading 114
6.6 Exercises and Software Demonstration 114

7 Gene Prediction 117
7.1 What is a Gene? 117
7.2 Computational Gene Finding 118
7.3 Measuring the Accuracy of Gene Predictions 121
7.4 *Ab initio* Methods: Searching for Signals and Content 124
7.4.1 Codon Usage 126
7.4.2 Finding Splice Sites with a Sequence Profile 126
7.4.3 Exon Chaining 131
7.5 Comparative Methods 134
7.5.1 General Remarks 134
7.5.2 Comparative Gene Prediction at the *Adh* Locus 135
7.6 Problems and Perspectives 138
7.7 Summary 139
7.8 Further Reading 139
7.9 Exercises 140

Part II Sequences in Time

8 Phylogeny 143
8.1 Is There a Tree?—Statistical Geometry 145
8.2 Likelihood-Mapping 146
8.3 The Number of Possible Phylogenies 148
8.4 Distance Methods 150
8.4.1 Average Linkage Clustering 152
8.4.2 Neighbor-Joining 155
8.5 Maximum Parsimony 157
8.6 Maximum Likelihood 159
8.7 Searching Through Tree Space 161
8.7.1 Nearest Neighbor Interchange 162
8.7.2 Subtree Pruning and Regrafting 163
8.7.3 Branch and Bound 163
8.8 Bootstrapping Phylogenies 164
8.9 Summary 166
8.10 Further Reading 167
8.11 Exercises and Study Questions 167

9 Sequence Variation and Molecular Evolution 169
9.1 The Record of Past Events 170
9.2 Mutations and Substitutions 171
9.3 The Molecular Clock 172
9.4 Explicit Models of Molecular Evolution 173
9.5 Estimating Evolutionary Rates 175
9.6 Coding Sequences: Synonymous and Non-Synonymous Substitutions 177
9.7 Substitutions in Globin Sequences 180
9.8 Applications of K_a/K_s 182
9.8.1 A Language Gene? 182
9.8.2 Selection in the Human Genome 183
9.9 Summary 184
9.10 Further Reading 185
9.11 Exercises 185

10 Genes in Populations: Forward in Time 187
10.1 Polymorphism and Genetic Diversity 187
10.2 The Neutral Theory 191
10.3 Modeling Evolution Forward in Time 193
10.4 The Neutral Wright-Fisher Model 194
10.4.1 Fixation and Loss of Alleles 195
10.4.2 The Hardy-Weinberg Law 197
10.4.3 Fixation Probability and Time to Fixation 197
10.4.4 Loss of Genetic Diversity 200
10.5 Adding Mutation to the Model 201
10.5.1 Finite Alleles Model 202
10.5.2 Infinite Alleles Model 203
10.5.3 Infinite Sites Model 203
10.6 Mutation Drift Balance 203
10.6.1 The Rate of Fixation 203
10.6.2 Number of Alleles 205
10.6.3 Genetic Diversity 207
10.7 Sampling Alleles from Populations 209
10.7.1 Ewens' Sampling Formula 209
10.7.2 Application 212
10.8 Selection 212
10.9 Summary 214
10.10 Further Reading 215
10.11 Exercises and Software Demonstration 215

11 Genes in Populations: Backward in Time 217
- 11.1 Individuals' Genealogies *vs.* Gene Genealogies 217
- 11.2 Forward *vs.* Backward in Time 218
- 11.3 The Coalescent 222
- 11.4 Coalescent *vs.* Phylogenetic Trees 224
- 11.5 The Infinite Sites Model and the Number of SNPs 224
- 11.6 Mathematical Properties of the Neutral Coalescent 225
  - 11.6.1 Tree Depth, Tree Size and the Number of Segregating Sites 225
  - 11.6.2 Heterozygosity 232
  - 11.6.3 The Distribution of Segregating Sites 233
- 11.7 Simulation Example 233
- 11.8 Recombination 233
- 11.9 Selection 237
- 11.10 Combining Recombination and Selection 238
- 11.11 Summary 242
- 11.12 Further Reading 242
- 11.13 Software Demonstrations and Exercises 242

12 Testing Evolutionary Hypotheses 245
- 12.1 Hudson-Kreitman-Aguadé (HKA) Test 245
- 12.2 Tajima's Test 248
- 12.3 Fu and Li's Test 251
- 12.4 McDonald-Kreitman Test 253
- 12.5 Minimum Number of Recombination Events 253
- 12.6 Detecting Linkage Disequilibrium 255
- 12.7 Implementations 257
- 12.8 Summary 257
- 12.9 Exercises and Software Demonstration 257

A `bioinformer` 259
- A.1 Alignment 259
  - A.1.1 Protein Substitution Matrices 259
  - A.1.2 Number of Alignments 261
  - A.1.3 Pairwise Alignment 262
- A.2 Match 263
  - A.2.1 String Matching 263
  - A.2.2 Suffix Tree 264
  - A.2.3 Repeat Searching 265
  - A.2.4 Hash Table 265
  - A.2.5 Dotplot 266
- A.3 Probability 266
  - A.3.1 Hidden Markov Model 267
- A.4 Evolution 268
  - A.4.1 Phylogeny 268
  - A.4.2 Drift 270

  A.4.3 Wright-Fisher 271
  A.4.4 Coalescent 273

B Probability 275

C Molecular Biology Figures and Tables 279

D Resources 285

Answers to Exercises 287

References 299

Glossary 313

Author Index 321

Subject Index 323

1

Introduction

> *The chromosome structures are at the same time instrumental in bringing about the development they foreshadow. They are law-code and executive power—or, to use another simile, they are architect's plan and builder's craft—in one.*
>
> *Erwin Schrödinger [220, p. 22]*

Since the discovery of chromosomal inheritance in the early 20-th century it has fascinated biologists that in contrast to, say, rocks or clocks, living organisms carry with them their own miniaturized part list, the genome. This specifies in its DNA sequence the primary structure of all proteins that make an organism as well as the primary structure of catalytically active RNA molecules. From 1995 onward, a rapidly growing number of genomes of free-living organisms have been sequenced in their entirety. This means that the succession of all bases, or base pairs (bp) since DNA is double stranded, of the organisms concerned is known. Figure 1.1 shows a sample of 106 organisms whose genomes have been sequenced. It contains representatives of all three domains of life, archebacteria (archaea), bacteria, and eucaryotes, which include humans. Organisms that are mentioned elsewhere in this book are marked by an arrow. There is a heavy bias toward bacteria, which is partly due to their medical importance. Another reason is that bacterial genomes are small, e.g. $1.8 \cdot 10^6$ bp for the human pathogen *Haemophilus influenzae*, and hence easier to sequence than for example the human genome, which is more than 1,000 times larger ($3.1 \cdot 10^9$ bp). Reading and writing of the genome form the molecular basis of life.

1.1 Reading and Writing

In every organism constant reading of the genome by the molecular genetic machinery of the cell is fundamental to sustaining its vital processes. The concomitant flow of information from DNA to proteins, but never in the reverse direction, is summarized by the well-known central dogma of molecular biology (Fig. 1.2).

Computational biology is traditionally concerned with two questions that build directly on the central dogma: where are the genes and what are their functions? In fact, these questions can be understood as an attempt to reproduce *in silico* the molecular genetic machinery of a cell. The enzymes making up this machinery locate specific genes with great precision. The subsequent expression of a particular gene takes place in a context of hundreds or even thousands of other genes active in that

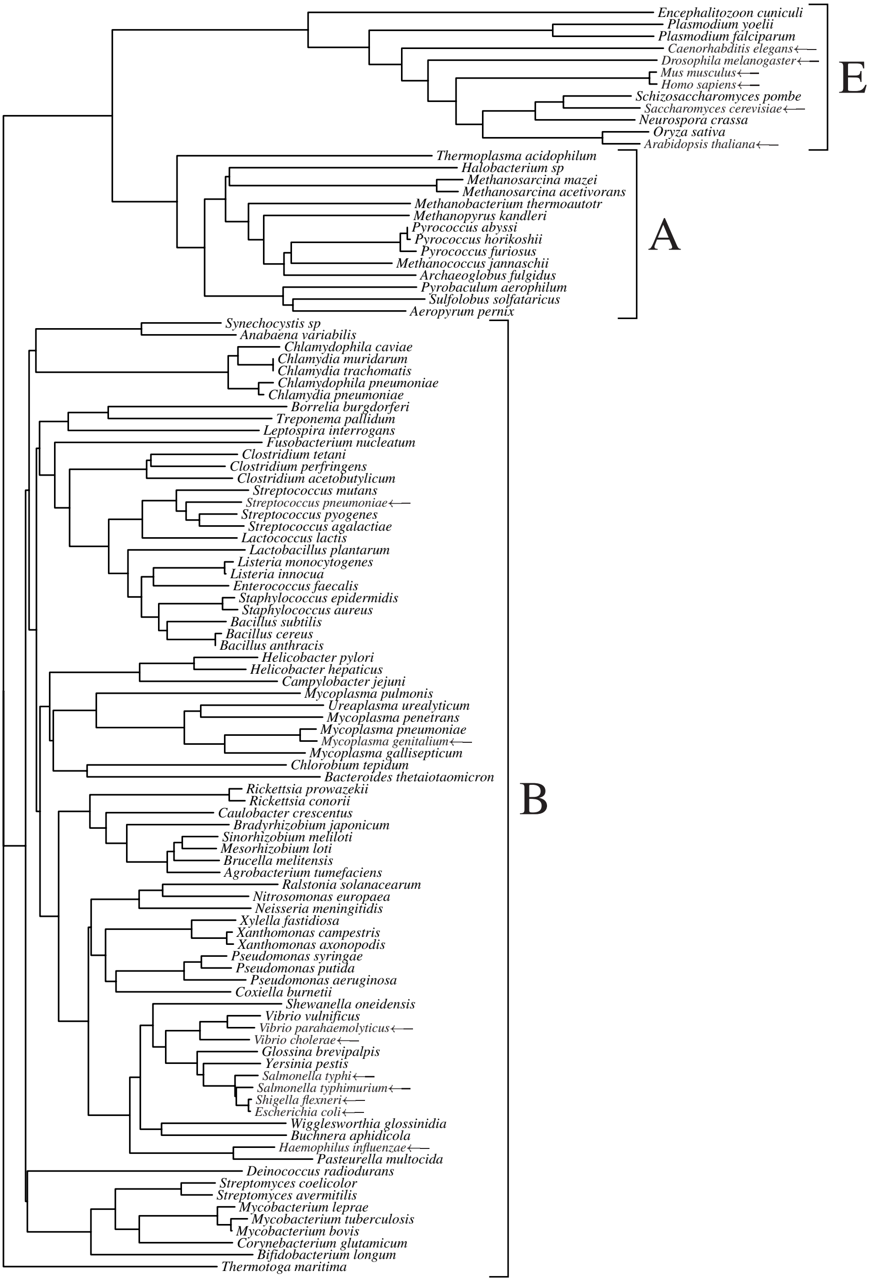

Encephalitozoon cuniculi
Plasmodium yoelii
Plasmodium falciparum
Caenorhabditis elegans←—
Drosophila melanogaster←—
Mus musculus←—
Homo sapiens←—
Schizosaccharomyces pombe
Saccharomyces cerevisiae←—
Neurospora crassa
Oryza sativa
Arabidopsis thaliana←—
E
Thermoplasma acidophilum
Halobacterium sp
Methanosarcina mazei
Methanosarcina acetivorans
Methanobacterium thermoautotr
Methanopyrus kandleri
Pyrococcus abyssi
Pyrococcus horikoshii
Pyrococcus furiosus
Methanococcus jannaschii
Archaeoglobus fulgidus
Pyrobaculum aerophilum
Sulfolobus solfataricus
Aeropyrum pernix
A
Synechocystis sp
Anabaena variabilis
Chlamydophila caviae
Chlamydia muridarum
Chlamydia trachomatis
Chlamydophila pneumoniae
Chlamydia pneumoniae
Borrelia burgdorferi
Treponema pallidum
Leptospira interrogans
Fusobacterium nucleatum
Clostridium tetani
Clostridium perfringens
Clostridium acetobutylicum
Streptococcus mutans
Streptococcus pneumoniae←—
Streptococcus pyogenes
Streptococcus agalactiae
Lactococcus lactis
Lactobacillus plantarum
Listeria monocytogenes
Listeria innocua
Enterococcus faecalis
Staphylococcus epidermidis
Staphylococcus aureus
Bacillus subtilis
Bacillus cereus
Bacillus anthracis
Helicobacter pylori
Helicobacter hepaticus
Campylobacter jejuni
Mycoplasma pulmonis
Ureaplasma urealyticum
Mycoplasma penetrans
Mycoplasma pneumoniae
Mycoplasma genitalium←—
Mycoplasma gallisepticum
Chlorobium tepidum
Bacteroides thetaiotaomicron
Rickettsia prowazekii
Rickettsia conorii
Caulobacter crescentus
Bradyrhizobium japonicum
Sinorhizobium meliloti
Mesorhizobium loti
Brucella melitensis
Agrobacterium tumefaciens
Ralstonia solanacearum
Nitrosomonas europaea
Neisseria meningitidis
Xylella fastidiosa
Xanthomonas campestris
Xanthomonas axonopodis
Pseudomonas syringae
Pseudomonas putida
Pseudomonas aeruginosa
Coxiella burnetii
Shewanella oneidensis
Vibrio vulnificus
Vibrio parahaemolyticus←—
Vibrio cholerae←—
Glossina brevipalpis
Yersinia pestis
Salmonella typhi←—
Salmonella typhimurium←—
Shigella flexneri←—
Escherichia coli←—
Wigglesworthia glossinidia
Buchnera aphidicola
Haemophilus influenzae←—
Pasteurella multocida
Deinococcus radiodurans
Streptomyces coelicolor
Streptomyces avermitilis
Mycobacterium leprae
Mycobacterium tuberculosis
Mycobacterium bovis
Corynebacterium glutamicum
Bifidobacterium longum
Thermotoga maritima
B

Fig. 1.1. Phylogeny showing the three domains of life: E: eucaryotes; A: archebacteria; B: bacteria. Computed from the ribosomal RNA sequences of 106 organisms whose genomes have been sequenced completely. Organisms mentioned in this book are marked by an arrow.

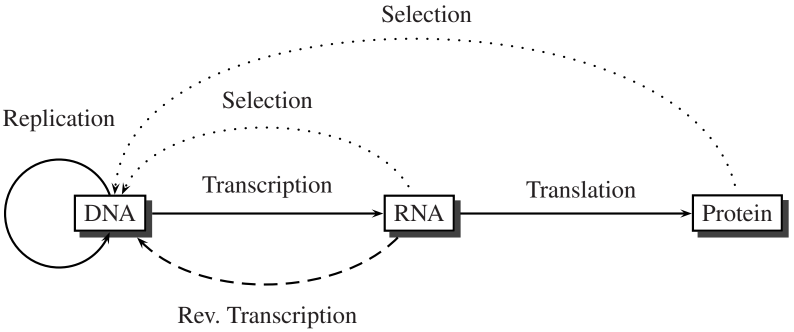


Fig. 1.2. Reading and writing at the machine level of life. Solid lines constitute the central dogma and correspond to "reading" from the genome. Dashed and dotted lines correspond to "writing" into the genome through reverse transcription and—on an evolutionary timescale—selection, respectively.

cell. It is this context of expression that constitutes to a large extent the function of a gene.

Given that there is enough information contained in a genome to largely determine the shape of a bacterial cell or of a human being, we might ask, how this information gets written into the genome. The answer is that some of it is written directly by reverse transcription (Fig. 1.2), which leads to the apparently excessive size of many eucaryotic genomes, including our own. However, more important for the preservation of useful information is a process that operates on a very different time scale from the molecular mechanisms mentioned so far: evolution by natural selection, which interacts with proteins and, perhaps to a lesser extent, transcripts (Fig. 1.2). The interaction between natural selection and these molecules is indirect: an organism carrying a protein that functions better than the original "wild type" leaves more offspring, who in turn leave even more offspring and as a result the advantageous information thrown up initially by a random mutation spreads throughout the population. It is this subtle yet intimate relationship between the reading aspect of the genome, which is the traditional subject of molecular biology and hence computational biology, and its writing aspect, usually the preserve of evolutionary biologists, which has motivated us to include evolutionary models in this book together with more classical sequence analysis methods.

1.2 Design and Scope of This Book

This book is divided in two parts. The first part deals with sequences in (sequence) space, the second with sequences in (evolutionary) time. We have also included a Glossary of important terms and Appendices on four topics:

1. Appendix A describes the Software `bioinformer`, which we have written to visualize some of the ideas treated in Parts I and II;
2. Appendix B briefly surveys important stochastic concepts that are used extensively in computational biology;
3. Appendix C deals with the chemical nature of nucleic acids and amino acids;
4. Appendix D contains pointers to software and databases useful in the practice of computational biology.

In addition, a web-page with supplementary material is posted at

`http://adenine.biz.fh-weihenstephan.de/icb/`

1.2.1 Sequences in Space

Amino acid and nucleotide sequences are often pictured as points in sequence space [173, 58], the metaphor underlying Part I. In sequence space individual sequences are ordered according to their distances measured in terms of the number of mutational steps necessary to get from one sequence to another. Let us start with the simplest case, a DNA "sequence" of length 1. There are four such sequences possible, {`A`, `C`, `G`, `T`}, and the corresponding sequence space can be visualized as a three-dimensional simplex or a tetrahedron (Fig. 1.3). Starting from any one of the nucleotides, we can reach all three others in a single step. The sequence space that accommodates all DNA sequences of length 2 consists of a tetrahedral hyperspace where each vertex of the hyper-tetrahedron is occupied by a simple tetrahedron. As before, the edges represent single mutational steps, i.e. the exchange of one residue by another. Figure 1.4 shows a projection of this space onto two dimensions. Each vertex in one of the four simple tetrahedrons is connected to three vertexes within the structure as well as to the corresponding vertexes in the other three tetrahedrons. For instance, vertex `AA` in the top tetrahedron is connected to `CA`, `TA`, and `GA` within the tetrahedron, as well as to `AG`, `AC`, and `AT` in the three other tetrahedrons. Each of the edges in this six-dimensional space has the same length.

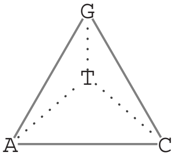


Fig. 1.3. Tetrahedral sequence space for DNA sequences of length 1.

In order to depict DNA sequences of length 3 in sequence space, we construct a hyper-tetrahedron with vertexes consisting of the structure depicted in Figure 1.4. Each sequence is connected to six other sequences within the basic tetrahedron as

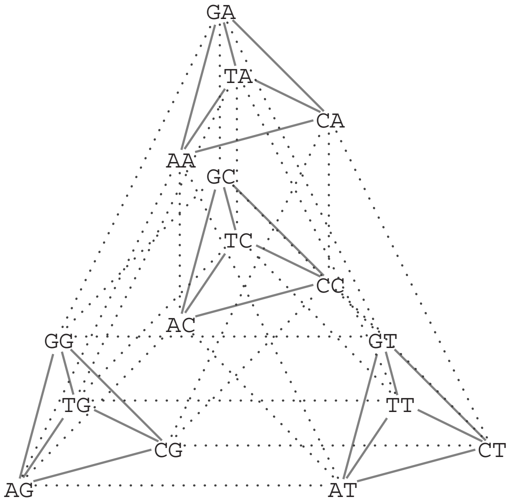


Fig. 1.4. Sequence space for DNA sequences of length 2.

well as to three other sequences at corresponding positions in the other three tetrahedrons. Figure 1.5 depicts the underlying nine-dimensional space, which represents the mutational distances between the 64 codon triplets.

Sequence space has two important properties: (i) it is highly connected and (ii) it has a high dimensionality. The connectedness means that within the space for sequences of length l the largest distance between any two points is l. The high dimensionality is a precondition for the connectedness. The number of dimensions necessary for constructing sequence space, d, is

$$d = (|\mathcal{A}| - 1)l,$$

where $|\mathcal{A}|$ is the size of the alphabet over which the sequence is formed. In our DNA example $|\mathcal{A}| = 4$ and hence the dimensionality of the codon space is $3 \cdot 3 = 9$, as we have already observed in Figure 1.5.

The sequence space describing all possible sequences of 1 kb, the length of a typical gene in *Escherichia coli*, contains $4^{1,000} \approx 10^{602}$ vertexes. In this vast space only a very small proportion of vertexes is occupied by actually existing sequences. A larger number of sequences, albeit still vanishingly few compared to the total number of possibilities, correspond to functional genes. Many different functions can be encoded in 1 kb. However, sequences with similar functions tend to cluster in sequence space, forming clouds of functional variants in a sea of lethal or as yet evolutionarily untested configurations.

The fact that close neighbors in sequence space tend to have similar functions motivates the sequence comparison methods discussed in Part I.

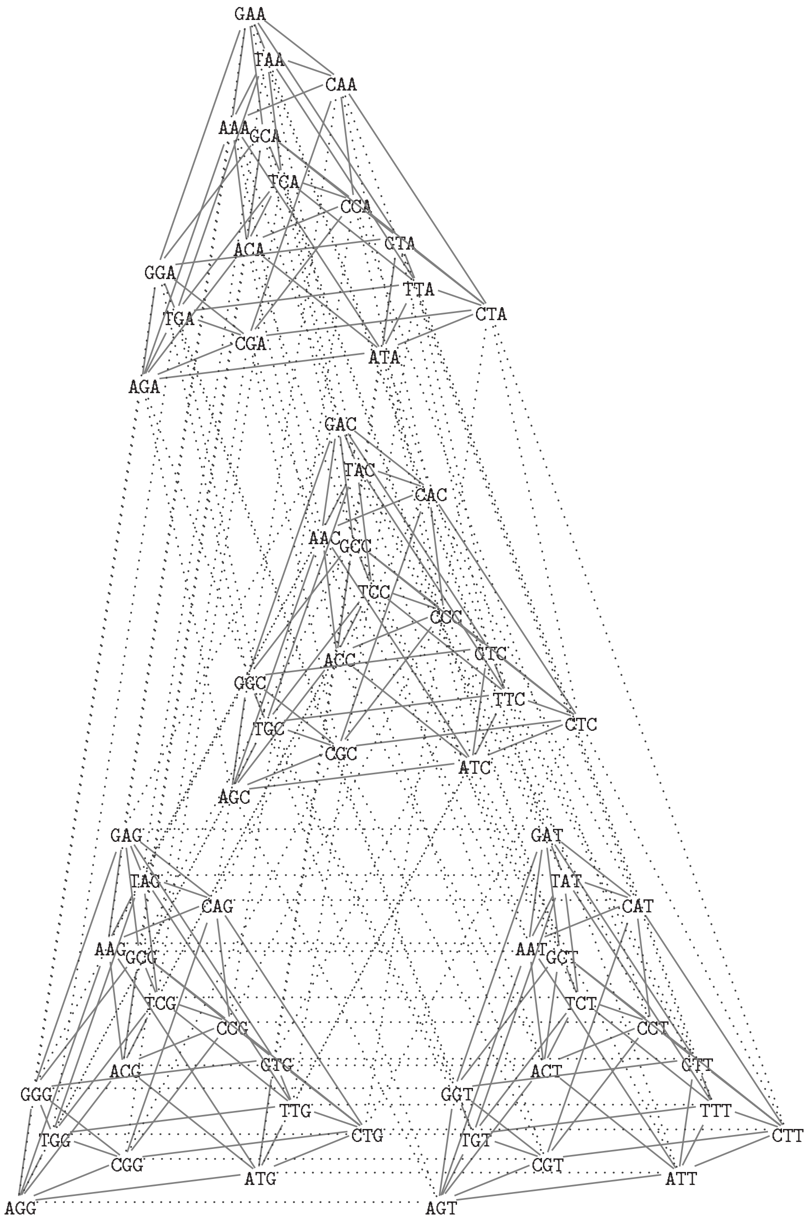

GAA
TAA
CAA
AAA
GCA
TCA
CCA
GTA
ACA
GGA
TTA
CTA
TGA
CGA
ATA
AGA
GAC
TAC
CAC
AAC
GCC
TCC
CCC
GTC
ACC
GGC
TTC
CTC
TGC
CGC
ATC
AGC
GAG
TAG
CAG
AAG
GCG
TCG
CCG
GTG
ACG
GGG
TTG
CTG
TGG
CGG
ATG
AGG
GAT
TAT
CAT
AAT
GCT
TCT
CCT
GTT
ACT
GGT
TTT
CTT
TGT
CGT
ATT
AGT

Fig. 1.5. DNA sequence space for DNA sequences of length 3, i.e. for all 64 codons.

We start off in Chapter 2 with the treatment of pairwise alignment using dynamic programming methods. These methods are effective but practical pairwise alignment algorithms reduce the time and memory requirement of dynamic programming by combining them with exact matching, the topic of Chapter 3. In Chapter 4 dynamic programming is wedded with exact matching to yield fast pairwise alignment algorithms. Pairwise alignments are a special case of multiple alignments, which are dealt with in Chapter 5. The information contained in a multiple sequence alignment can be used to classify protein sequences into protein families, of which there are far fewer than actual protein sequences. The tools used for this classification, profile analysis and hidden Markov models, are the topics of Chapter 6. Alignments, profiles, as well as hidden Markov models, are used in various combinations to predict the location of genes in as yet unannotated DNA sequences. Our introduction to gene prediction in Chapter 7 concludes Part I.

1.2.2 Sequences in Time

Part II is concerned with sequence evolution. Sequences change over time through mutation, which corresponds to a random walk in sequence space. Given the sheer infinite size of this space, it may come as a surprise that anything as useful as the set of instructions specifying the molecular parts of a human being could ever be found therein. However, we have already explained that in principle few mutational steps are necessary to get from any given sequence to any other. In addition, selection provides a powerful signpost at every vertex in sequence space, thereby guiding the mutational random walk. The condition for the evolvability of a sequence was made precise in 1970 by the English evolutionary biologist Maynard Smith [173]. He noted that for evolution by natural selection to occur, sequence space needs to be traversable in unit mutational steps without passing through non-functional intermediates. We have already seen that a sequence of length l is embedded in an $d = (|\mathcal{A}| - 1)l$-dimensional space. In other words, it has d neighbors that can be reached by a single mutational step. The condition for evolution by natural selection is that for any given functional sequence there exists at least one functional one-mutation neighbor. If we call f the fraction of viable single-step mutations a given sequence can change into, the condition for evolvability simply amounts to [173]

$$f \cdot d \geq 1.$$

Since many mutations are known to have no effect on fitness, i.e. to be neutral, this condition will often be met [221].

Chapter 8 introduces what is perhaps the best-known aspect of sequence evolution, the reconstruction of phylogenies. Such reconstruction is based on the differences between homologous sequences observed in a multiple sequence alignment. These differences are the topic of Chapter 9. In particular, we shall be interested in differences found between protein-coding sequences, as these can help us detect selection, the force that keeps sequences on the straight and narrow in sequence space. The analysis of phylogenies as well as the analysis of differences in coding sequences are usually based on inter-species comparisons. In Chapter 10 we zoom in on a given species and concentrate on intra-species sequence comparisons. To be more precise, we shall concentrate on the dynamics of genes found in one population of organisms. The traditional method to trace the history of genes in populations moves forward in time and this is also the perspective we start off with in Chapter 10. A reverse in time perspective on evolution was developed in the late 1970s and early 1980s. Today it is the standard way to think about genes in populations and Chapter 11 introduces the corresponding methodology, known as coalescent theory. One of the central concerns of any investigation of genes in time is the detection of selection. Without it, there would be no function, that is no phenotype attached to the sequence genotype. Chapter 12 makes this concern explicit by describing a number of formal hypothesis tests that can be applied to sequence data in order to detect selection.

Part I

Sequences in Space

2

Optimal Pairwise Alignment

> *So now we are going to sequence the human genome and then we will sequence the chimp genome. Once this is done, we align each gene from the two organisms and the gene that is left over must be the language gene. If humans have it, we call it the Chomsky gene. If chimps have it, we call it the Chimsky gene.*
>
> *Sydney Brenner [26]*

Living organisms are the product of an evolutionary history. This has important consequences for the practice of biology, whether of the organismal or of the molecular kind. In classical biology, organs that derive from a recent common ancestor are called homologous. For instance, a human hand is homologous to a chimpanzee's hand, a cat's paw, and a bat's wing. These structures are characterized by common anatomical features and similar, albeit highly diverse, functions. In contrast, the wings of bats and flies are not homologous and anatomically very different, even though they are both organs of flight. Such organs are said to be analogous. To put matters simply, homology tends to imply similarity of function, while similarity of function may be due either to analogy or to homology.

In molecular biology the search for homologous traits, in this case residues in DNA or protein sequences, reappears in the guise of the alignment problem. As explained in detail below, two sequences are aligned by writing homologous residues on top of each other. This may or may not work well, i.e. the resulting alignment may or may not be convincing. If the two proteins can be aligned reliably, they tend to be similar. Moreover, they tend to have similar functions. The fact that similar sequences encode similar functions is the reason why we start this book with alignment algorithms: such algorithms are used for finding similar sequences of known function and thereby efficiently determine to a first approximation the function of novel genes.

Alignments can be either pairwise, comprising just two sequences, or multiple, comprising an arbitrary number of sequences. Pairwise alignments might be regarded as a special case of multiple alignments. In practice, however, the computational complexity of aligning multiple sequences is such that the corresponding algorithms are usually not straight extensions of the pairwise approaches. Instead, multiple alignments are often constructed by repeatedly merging pairwise alignments. In this chapter we concentrate on basic pairwise alignments, while multiple alignments are the topic of Chapter 5.

There are numerous approaches to aligning two sequences, but three of the most important strategies are known as *global*, *local*, and *overlap* (Fig. 2.1). A global alignment algorithm is appropriate if two sequences are homologous across their entire length. However, many sequences only share homologous regions while other parts of the molecule are quite unrelated. Sequence regions that correspond to a functional unit are also known as *domains*. For example, an important class of developmental genes, the *Hox* genes, were first discovered in the fruit fly *Drosophila melanogaster*. There they act as transcriptional regulators that determine the identity of body segments along the anterior-posterior axis of the developing embryo. *D. melanogaster* has eight *Hox* genes located on chromosome 3. The remarkable thing about these genes is that their layout along the chromosome corresponds to the layout of larval proto-segments in which they are most highly expressed. Today, *Hox* genes have been discovered in almost all animals. In mammals, for example, they occur in four complexes, which each correspond to one of the *Drosophila Hox* clusters. Again, the polarity of the genes' layout along the mammalian chromosomes reflects the anterior-posterior ordering of their expression in the early embryo. All *Hox* genes are characterized by a conserved region of 60 amino acids, the homeodomain, which binds to specific regulatory motifs on the DNA. Figure 2.2 displays a global alignment of the two *Hox* proteins from human and *D. melanogaster* that are expressed in the most anterior segments of the embryo. Homologous amino acids are written on top of each other, deleted amino acids are indicated by gaps (−). Notice that a deletion in one sequence implies an insertion in the other. Hence insertions and deletions are often referred to as "indels".

As shown in Figure 2.2, the annotated homeodomain has remained highly conserved over the approximately 590 million years [45, p. 389] that separate the protostomes, to which *D. melanogaster* belongs, from the vertebrates. In contrast, regions outside the homeodomain have changed beyond recognition. A local alignment algorithm is appropriate for the discovery of specific conserved regions such as homeodomains.

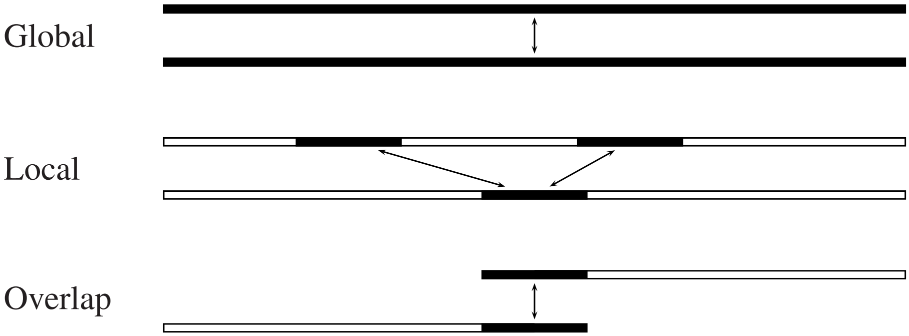
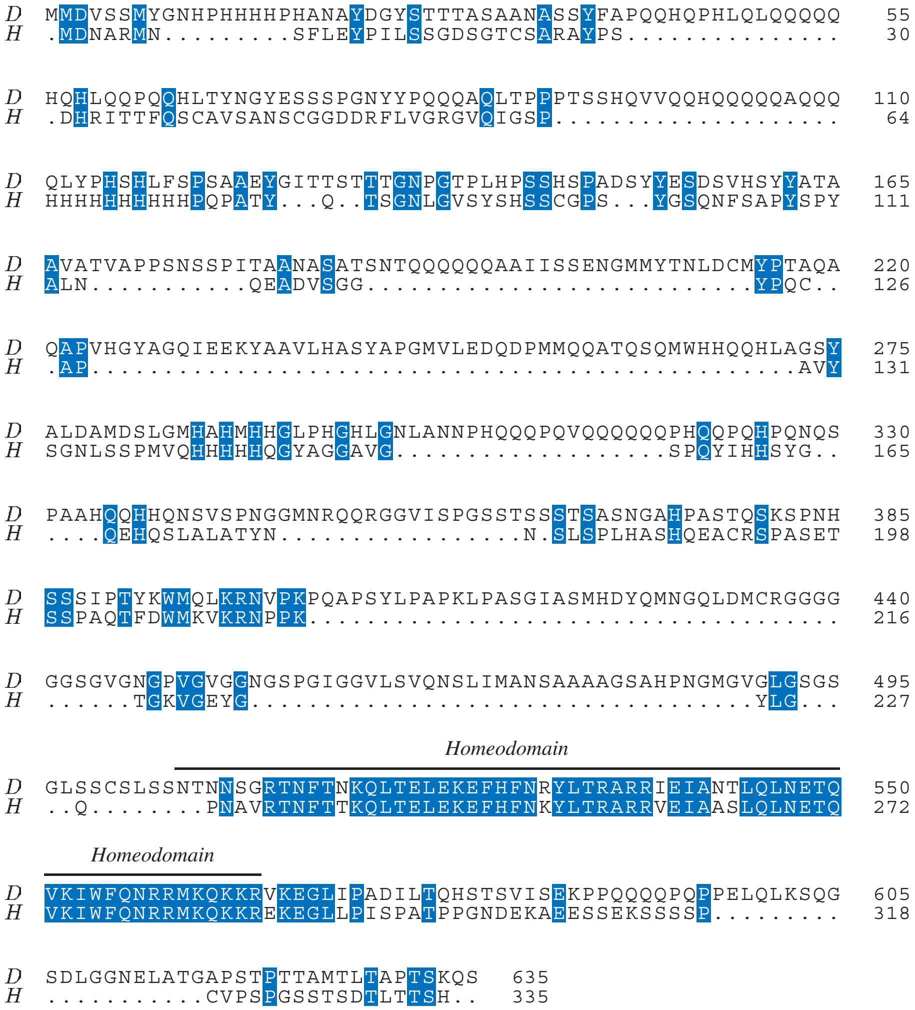


Fig. 2.1. Types of pairwise alignment discussed in this chapter. Homologous regions are indicated in black.

Overlap alignments, finally, are used to align sequences where only the ends match. This is the case for example in the context of the assembly of sequences from a genome sequencing project.



```
D  MMDVSSMYGNHPHHHHPHANAYDGYSTTTASAANASSYFAPQQHQPHLQLQQQQQ   55
H  .MDNARMN.........SFLEYPILSSGDSGTCSARAYPS...............   30

D  HQHLQQPQQHLTYNGYESSSPGNYYPQQQAQLTPPPTSSHQVVQQHQQQQQAQQQ  110
H  .DHRITTFQSCAVSANSCGGDDRFLVGRGVQIGSP....................   64

D  QLYPHSHLFSPSAAEYGITTSTTTGNPGTPLHPSSHSPADSYYESDSVHSYYATA  165
H  HHHHHHHHHHPQPATY...Q..TSGNLGVSYSHSSCGPS...YGSQNFSAPYSPY  111

D  AVATVAPPSNSSPITAANASATSNTQQQQQQAAIISSENGMMYTNLDCMYPTAQA  220
H  ALN...........QEADVSGG...........................YPQC..  126

D  QAPVHGYAGQIEEKYAAVLHASYAPGMVLEDQDPMMQQATQSQMWHHQQHLAGSY  275
H  .AP.................................................AVY  131

D  ALDAMDSLGMHAHMHHGLPHGHLGNLANNPHQQQPQVQQQQQQPHQQPQHPQNQS  330
H  SGNLSSPMVQHHHHHQGYAGGAVG...................SPQYIHHSYG..  165

D  PAAHQQHHQNSVSPNGGMNRQQRGGVISPGSSTSSSTSASNGAHPASTQSKSPNH  385
H  ....QEHQSLALATYN.................N.SLSPLHASHQEACRSPASET  198

D  SSSIPTYKWMQLKRNVPKPQAPSYLPAPKLPASGIASMHDYQMNGQLDMCRGGGG  440
H  SSPAQTFDWMKVKRNPPK.....................................  216

D  GGSGVGNGPVGVGGNGSPGIGGVLSVQNSLIMANSAAAAGSAHPNGMGVGLGSGS  495
H  ......TGKVGEYG...................................YLG...  227

                                 Homeodomain
                  ____________________________________________
D  GLSSCSLSSNTNNSGRTNFTNKQLTELEKEFHFNRYLTRARRIEIANTLQLNETQ  550
H  ..Q........PNAVRTNFTTKQLTELEKEFHFNKYLTRARRVEIAASLQLNETQ  272

      Homeodomain
   _______________
D  VKIWFQNRRMKQKKRVKEGLIPADILTQHSTSVISEKPPQQQQPQPPELQLKSQG  605
H  VKIWFQNRRMKQKKREKEGLLPISPATPPGNDEKAEESSEKSSSSP.........  318

D  SDLGGNELATGAPSTPTTAMTLTAPTSKQS  635
H  ...........CVPSPGSSTSDTLTTSH..  335
```

Fig. 2.2. Global alignment of two orthologous *Hox* proteins with annotated homeodomain. *D*: homeotic labial protein (*Lab*) from *Drosophila melanogaster*; *H*: homeobox protein *Hox-A1* from human; white on gray: identical amino acids; black on gray: similar amino acids; –: gaps, i.e sequence insertions/deletions.

Whether global, local, or overlap, alignments can be computed by two types of algorithms: optimal and heuristic. Historically, optimal alignment algorithms were developed first (Table 2.1) and are based on a computational technique known as "dynamic programming". This guarantees that the best possible alignment is found given the parameters chosen. Heuristic algorithms, on the other hand, can be thought of as fast procedures for approximating their optimal counterparts. Heuristic approaches to alignment are the topic of Chapter 4. The first optimal alignment algorithm was published in 1970 [189], while the first widely used heuristic alignment algorithm, the FASTA algorithm, was published over a decade later [197].

Table 2.1. Landmarks in the development of pairwise alignment algorithms.

Year	Authors	Algorithm	Approach
1970	Needleman & Wunsch	Global	Optimal
1981	Smith & Waterman	Local	Optimal
1988	Pearson & Lipman	FASTA	Heuristic
1990	Altschul *et al.*	BLAST	Heuristic
1997	Altschul *et al.*	Gapped BLAST	Heuristic

In this chapter we explain how dynamic programming is used to compute global, local, and overlap alignments (Fig. 2.1). We begin by clarifying further what is meant by an alignment.

2.1 What Is an Alignment?

DNA sequences and the protein sequences they encode change in evolutionary time through mutation. The simplest types of mutation are point mutations and insertions/deletions, also known as *indels*. These two types of change are modeled by classical alignment algorithms. Say we wish to align the two DNA sequences AACGT and ACCGTT. This is done by writing their residues on top of each other such that either two residues are paired, corresponding to presence or absence of a point mutation, or a residue is paired with a gap, corresponding to an insertion or deletion. It is not permitted to pair a gap with a gap. The reason for this is not that two sequences cannot inherit the same deletion, they certainly can. However, there is not enough information to infer such a course of evolution from just two sequences. Figure 2.3 displays six example alignments that can be generated by following these rules. The question is, which of the possible alignments is the "best"?

1	2	3	4	5	6
AACGT- ACCGTT	-AACGT ACCGTT	A-ACGT ACCGTT	AACGT------ -----ACCGTT	AA-CGT- A-CCGTT	-A-A-C-G-T- A-C-C-G-T-T

Fig. 2.3. Six of the 3,653 possible ways to align the DNA sequences AACGT and ACCGTT.

Intuitively, we might prefer alignment 1 with its four matched residues to alignment 2, which contains only a single matched residue. However, alignment 5 also contains four matches. On the other hand, it contains three gaps, whereas alignment 1 contains only one gap, leaving us still with the feeling that alignment 1 is best. How can we turn this intuition into an objective evaluation procedure?

2.2 Biological Interpretation of the Alignment Problem

In order to choose the "best" alignment, we need to recall that, biologically speaking, we seek to align homologous residues. In addition, we assume that evolution is parsimonious. In other words, when calculating an alignment we aim to minimize the number of evolutionary changes implied by an alignment.

Consider again the alignments in Figure 2.3. The first one implies that since divergence from their last common ancestor the two sequences have undergone one point mutation and one insertion/deletion. Hence, the alignment implies two evolutionary events. In contrast, the second alignment implies five such events: one insertion/deletion and four point mutations. Therefore, we would prefer the first alignment to the second.

Instead of *minimizing* the number of evolutionary events, it is conceptually equivalent to *maximize* a score that reflects the similarity of the two sequences. All alignment algorithms surveyed in this chapter are based on such maximizing scoring schemes and we will explain why this is so later on in Section 2.10.

2.3 Scoring Alignments

The minimal unit of an alignment consists of one position from each of the two sequences being compared. When scoring such a pair of positions it is customary to distinguish between pairs consisting of two residues and pairs that form part of an indel.

As to indels, the simplest way to model them is known as the *linear gap model*. Under this model the score of a gap, G, is defined as

$$G = l \cdot g_{\mathrm{e}},$$

where l is the length of the gap and and g_{e} the gap extension cost. However, molecular biologists have known for a long time that gaps often extend over several residues. In other words, the scoring of the existence of an indel needs to be distinguished from the scoring of its length. This idea is embodied in the widely used *affine gap model*, according to which the score for a gap is computed as

$$G = g_{\mathrm{o}} + l \cdot g_{\mathrm{e}}, \tag{2.1}$$

where g_{o} denotes gap opening. A reasonable combination of parameters could be $g_{\mathrm{o}} = -5$ and $g_{\mathrm{e}} = -2$, i.e., gap opening is penalized more than twice as heavily as gap extension [10].

When scoring residues, it depends on whether nucleotides or amino acids are considered. Nucleotides are often modeled as all having the same rate of mutation. This is a simplification, as transitions (purine → purine or pyrimidine → pyrimidine) are usually more frequent than transversions (pyrimidine ⇌ purine), but in practice this model works quite well. For simplicity we distinguish here only between matched and mismatched nucleotides. A mismatch might be scored as -3 and a match as +1 [10].

In contrast to nucleotides, different amino acids mutate at very different rates. To be more precise, it is the codons specifying the amino acid sequence that mutate and it is clear that the number of mutational steps necessary for converting one amino acid into another can range from one to three (cf. Table C.4). This variation in mutational distance is one reason for the different mutation rates among amino acids. More important, however, are their highly diverse physico-chemical properties (Fig. C.4), which result in substitutions that disrupt protein structure to very different degrees.

Whatever the underlying molecular causes might be, the fact of differing amino acid mutation rates is routinely described in terms of substitution matrices. These give scores for all $20^2 = 400$ possible changes between amino acids and are discussed next.

2.4 Amino Acid Substitution Matrices

Today, there are far more nucleotide than protein sequence data available. This reflects the fact that genomes are bigger than the proteomes they encode. However, protein sequencing started before the discovery of the double helical structure of DNA in 1953. Between 1949 and 1955 the first complete protein sequence, that of bovine insulin, was determined by Fred Sanger and coworkers [129, p. 212]. It was only once Sanger and Maxam and Gilbert independently devised simple methods for DNA sequencing in the mid 1970s that an organism's nucleotide genotype became much more accessible than its protein phenotype. For the early development of alignment algorithms proteins were therefore the focus of research.

Molecular biologists often talk about "mutations in protein sequences". This is shorthand for saying that the sequence encoding the protein has mutated. Due to the degenerate nature of the genetic code, the mutational distance between amino acids varies from one to three steps. Figure 2.4 shows the 20 proteinogenic amino acids in their codon space, which was already displayed in Figure 1.5. Consider, for example the amino acid methionine (`Met`), which is encoded by a single codon, `ATG`. Threonine (`Thr`), which is encoded by the four codons $\{\texttt{ACA}, \texttt{ACC}, \texttt{ACG}, \texttt{ACT}\} \equiv$ `AC[ACGT]`, is separated by one mutational step from methionine. Glycine (`Gly`), which is encoded by `GG[ACGT]`, is two mutations removed. Histidine (`His`, `CA[TC]`), finally, is an example of an amino acid that can only be reached by changing all three nucleotides encoding methionine.

Apart from the differences in mutational distances separating the amino acids, they also differ widely in their physico-chemical properties. Figure C.4 shows that

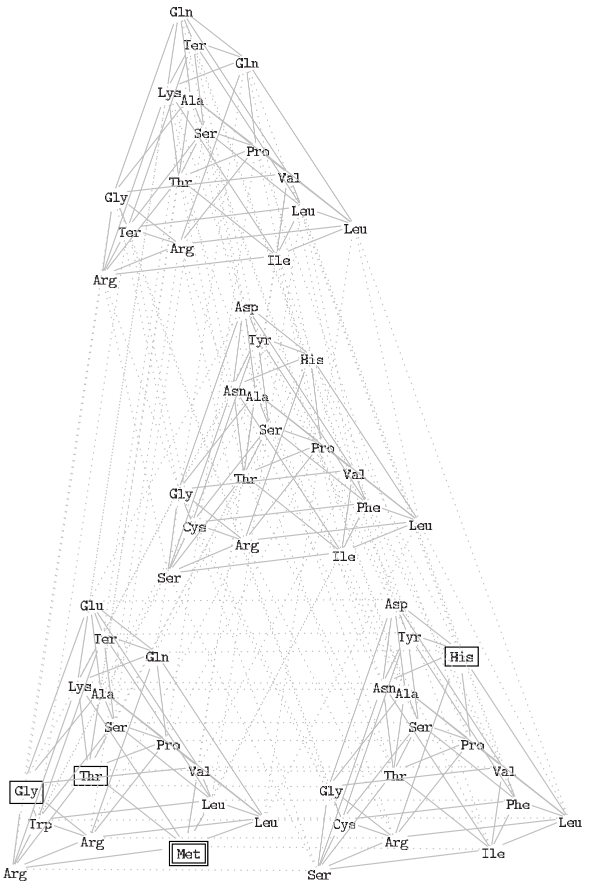

Gln
Ter
Gln
Lys
Ala
Ser
Pro
Thr
Val
Gly
Leu
Leu
Ter
Arg
Ile
Arg
Asp
Tyr
His
Asn
Ala
Ser
Pro
Thr
Val
Gly
Phe
Cys
Leu
Arg
Ile
Ser
Glu
Ter
Gln
Lys
Ala
Ser
Pro
Thr
Val
Gly
Leu
Trp
Leu
Arg
Met
Arg
Asp
Tyr
His
Asn
Ala
Ser
Pro
Thr
Val
Gly
Phe
Leu
Cys
Arg
Ile
Ser

Fig. 2.4. The 20 proteinogenic amino acids in codon space (Fig. 1.5). Starting from `Met`, `Thr` can be reached in one mutational step in the underlying codon, `Gly` in two steps and `His` in three.

one half of amino acid side chains are classified as polar, the other half as non-polar. Moreover, the size of the side chains ranges from tiny in glycine, where it consists of a single hydrogen atom, to bulky in tryptophan, with its aromatic two-ring side chain. Additional physico-chemical categories that distinguish the twenty proteinogenic amino acids include negative and positive charge and aliphatic *vs.* non-aliphatic. Given the diversity of amino acids in mutational distance as well as chemical properties, a simple match/mismatch scoring scheme is not sufficient.

There is an additional fundamental aspect of biological sequences that the match/mismatch scheme ignores: evolutionary time. Two sequences that are being compared might have diverged for only a few thousand years. In this case a homologous pair of residues is less likely to be mismatched than after tens of millions of years. Substitution matrices are designed to take both the diversity of amino acids, as well as the evolutionary dimension of sequence comparison into account. Two series of amino acid scoring matrices are in common use today, the PAM and the BLOSUM series.

2.4.1 PAM Matrices

The first widely used scoring scheme for aligned amino acids was devised in the 1970s by Margret Dayhoff and coworkers [46]. In her model of protein evolution Dayhoff incorporated the observation that pairs of amino acids mutate at different rates, i.e. are differentially conserved. For instance, as shown in Table 2.2, lysine (`K`) is 37 times more likely to mutate into its physico-chemical cousin arginine (`R`) than into, say, the chemically more different leucine (`L`, Fig. C.4). Apart from physico-chemical similarity, the rate of amino acid mutation depends on the divergence time of the sequences concerned. In the limit of zero divergence time, two sequences are identical and the probabilities of any mutations are all zero. This would correspond to all entries in Table 2.2 being zero except for the values on the main diagonal, which would be 10,000, i.e. correspond to a probability of 1. In contrast, in the limit of infinite divergence time, any amino acid is equally likely to have mutated into any other. In this case all entries in Table 2.2 would be equal to the background frequencies of the amino acids labeling the rows. Therefore, a time-graded series of substitution matrices is necessary for aligning pairs of protein sequences that are separated by different divergence times.

Dayhoff's starting point for accounting for sequence conservation over time were alignments of closely related proteins from which all possible $20^2 = 400$ mutation probabilities were computed. These probabilities were normalized such that they corresponded to a change in 1% of amino acid positions (Fig. 2.5). In other words, the original probabilities were normalized in order to correspond uniformly to comparisons of sequences separated by 1 **P**ercent **A**ccepted **M**utations, or 1 PAM. "Accepted" refers to the fact that these mutations have passed the filter of selection, as opposed to those mutations that never made it into an extant population.

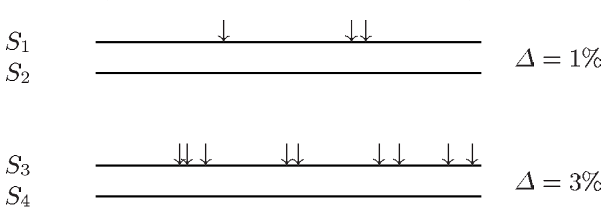


Fig. 2.5. Pairs of aligned protein sequences differ to varying degrees. Hence the need to normalize the mutation probabilities computed from such alignments. Δ: difference; $\downarrow$: mutation.

Table 2.2. Matrix of amino acid mutation probabilities (times 10,000), also known as a state transition matrix. Large numbers on the diagonal indicate a high probability of no change. For example, the probability of an alanine mutating into an arginine (`A`→`R`) is $M_{\mathtt{AR}} = 2 \cdot 10^{-4}$, while the reverse mutation probability (`R`→`A`) is $M_{\mathtt{RA}} = 10^{-4}$. Data taken from [46].

	A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W	Y	V
A	9867	2	9	10	3	8	17	21	2	6	4	2	6	2	22	35	32	0	2	18
R	1	9913	1	0	1	10	0	0	10	3	1	19	4	1	4	6	1	8	0	1
N	4	1	9822	36	0	4	6	6	21	3	1	13	0	1	2	20	9	1	4	1
D	6	0	42	9859	0	6	53	6	4	1	0	3	0	0	1	5	3	0	0	1
C	1	1	0	0	9973	0	0	0	1	1	0	0	1	0	0	5	1	0	3	2
Q	3	9	4	5	0	9876	27	1	23	1	3	6	4	0	6	2	2	0	0	1
E	10	0	7	56	0	35	9865	4	2	3	1	4	1	0	3	4	2	0	1	2
G	21	1	12	11	1	3	7	9935	1	0	1	2	1	1	3	21	3	0	0	5
H	1	8	18	3	1	20	1	0	9912	0	1	1	0	2	3	1	1	1	4	1
I	2	2	3	1	2	1	2	0	0	9872	9	2	12	7	0	1	7	0	1	33
L	3	1	3	0	0	6	1	1	4	22	9947	2	45	13	3	1	3	4	2	15
K	2	37	25	6	0	12	7	2	2	4	1	9926	20	0	3	8	11	0	1	1
M	1	1	0	0	0	2	0	0	0	5	8	4	9874	1	0	1	2	0	0	4
F	1	1	1	0	0	0	0	1	2	8	6	0	4	9946	0	2	1	3	28	0
P	13	5	2	1	1	8	3	2	5	1	2	2	1	1	9926	12	4	0	0	2
S	28	11	34	7	11	4	6	16	2	2	1	7	4	3	17	9840	38	5	2	2
T	22	2	13	4	1	3	2	2	1	11	2	8	6	1	5	32	9871	0	2	9
W	0	2	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	9976	1	0
Y	1	0	3	0	3	0	1	0	4	1	1	0	0	21	0	1	1	2	9945	1
V	13	2	1	1	3	2	2	3	3	57	11	1	17	1	3	2	10	0	2	9901

The divergence time implied by 1 PAM varies strongly depending on the set of organisms we are looking at. In a comparison of 32 proteins from *D. melanogaster* and *D. obscura*, 1.91 non-synonymous substitutions were found per non-synonymous site per 10^9 years [168, p. 191]. Since there are approximately two non-synonymous sites per amino acid, this implies that 1 PAM corresponds to

$$\frac{10^9}{1.91 \cdot 2 \cdot 100} \approx 2.62$$

million years. However, in the hominoid lineage substitution rates are lower. A comparison of 97 protein coding genes from human and chimpanzee uncovered 0.006 non-synonymous substitutions per non-synonymous site [261]. Assuming human/chimpanzee divergence time of 5.5 million years, 1 PAM corresponds to

$$\frac{5.5 \cdot 10^6}{0.006 \cdot 2 \cdot 100} \approx 4.58$$

million years in this case, 1.75 times more than in *Drosophila*.

For sequences diverged by 1 PAM, Table 2.2 shows the probabilities with which each of the 20 amino acids mutated into any of the other 19 amino acids or remained unchanged. In order to obtain the probability matrix appropriate for an evolutionary distance of n PAM, the original probability matrix only needs to be taken to the power of n. In this way substitution probabilities for arbitrary evolutionary distances can be computed.

Notice the distinction between the number of mutations a stretch of 100 amino acids has accepted, and the percent difference between sequences. For example, an evolutionary distance of 160 PAM happens to correspond to an average difference of 70% (Fig. 2.6). In order to calculate this, let M_{ij} be the probability of amino acid i mutating into amino acid j over an evolutionary distance of 160 PAM. The corresponding percent difference is computed as

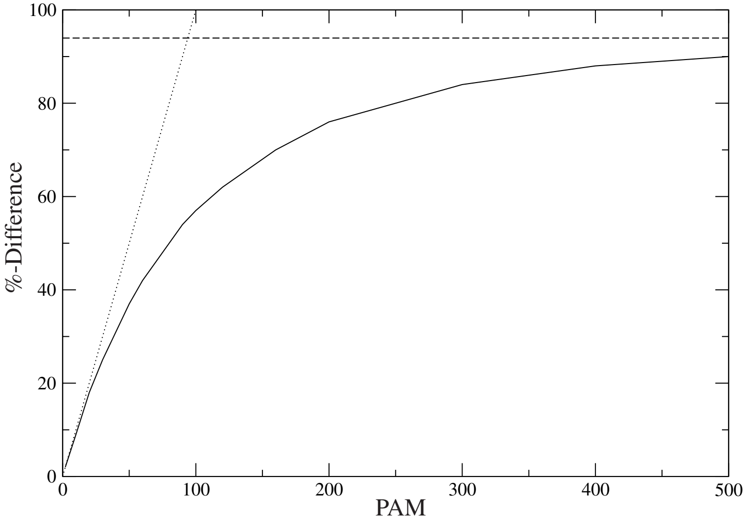


Fig. 2.6. Expected %-difference between two protein sequences as a function of their evolutionary distance expressed in units of PAM (solid line). Dotted line: equality between PAM number and %-difference; Dashed line: asymptotic %-difference for infinite PAM.

$$\left(1-\sum_{i}^{20} M_{ii}\cdot p_i\right)\cdot 100,$$

were p_i is the frequency of the i-th amino acid in the data set from which the original PAM matrix was constructed. This is also referred to as the background frequency of the amino acid. The percent difference will always be less or equal to the PAM number, because mutations may affect the same amino acid more than once, which is increasingly likely to happen with growing PAM numbers. Notice also that the percent difference between two sequences must lie between 0% and 100%, while the number of mutations a sequence has accepted varies between 0 and infinity.

However, a PAM matrix like PAM160 depicted in Table 2.3, contains scores rather than probabilities. The substitution probabilities appropriate for an evolutionary distance of n PAM are converted into scores in two steps. First they are divided by the background frequency of the original amino acid. For example, the probability of mutating from alanine to arginine (`A`→`R`), $M_{\mathtt{AR}}$, is divided by the background frequency of `A`, $p_{\mathtt{A}}$, while the probability of mutating in the reverse direction (`R`→`A`), $M_{\mathtt{RA}}$, is divided by the background frequency of `R`, $p_{\mathtt{R}}$. Thus for any pair of amino acids i, j, we define the ratio R_{ij}:

Table 2.3. PAM160 amino acid substitution matrix. Match scores are shown in bold.

	A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W	Y	V
A	**2**	-2	0	0	-2	-1	0	1	-2	-1	-2	-2	-1	-3	1	1	1	-5	-3	0
R	-2	**6**	-1	-2	-3	1	-2	-3	1	-2	-3	3	-1	-4	-1	-1	-1	1	-4	-3
N	0	-1	**3**	2	-4	0	1	0	2	-2	-3	1	-2	-3	-1	1	0	-4	-2	-2
D	0	-2	2	**4**	-5	1	3	0	0	-3	-4	0	-3	-6	-2	0	-1	-6	-4	-3
C	-2	-3	-4	-5	**9**	-5	-5	-3	-3	-2	-6	-5	-5	-5	-3	0	-2	-7	0	-2
Q	-1	1	0	1	-5	**5**	2	-2	2	-2	-2	0	-1	-5	0	-1	-1	-5	-4	-2
E	0	-2	1	3	-5	2	**4**	0	0	-2	-3	-1	-2	-5	-1	0	-1	-7	-4	-2
G	1	-3	0	0	-3	-2	0	**4**	-3	-3	-4	-2	-3	-4	-1	1	-1	-7	-5	-2
H	-2	1	2	0	-3	2	0	-3	**6**	-3	-2	-1	-3	-2	-1	-1	-2	-3	0	-2
I	-1	-2	-2	-3	-2	-2	-2	-3	-3	**5**	2	-2	2	0	-2	-2	0	-5	-2	3
L	-2	-3	-3	-4	-6	-2	-3	-4	-2	2	**5**	-3	3	1	-3	-3	-2	-2	-2	1
K	-2	3	1	0	-5	0	-1	-2	-1	-2	-3	**4**	0	-5	-2	-1	0	-4	-4	-3
M	-1	-1	-2	-3	-5	-1	-2	-3	-3	2	3	0	**7**	0	-2	-2	-1	-4	-3	1
F	-3	-4	-3	-6	-5	-5	-5	-4	-2	0	1	-5	0	**7**	-4	-3	-3	-1	5	-2
P	1	-1	-1	-2	-3	0	-1	-1	-1	-2	-3	-2	-2	-4	**5**	1	0	-5	-5	-2
S	1	-1	1	0	0	-1	0	1	-1	-2	-3	-1	-2	-3	1	**2**	1	-2	-3	-1
T	1	-1	0	-1	-2	-1	-1	-1	-2	0	-2	0	-1	-3	0	1	**3**	-5	-3	0
W	-5	1	-4	-6	-7	-5	-7	-7	-3	-5	-2	-4	-4	-1	-5	-2	-5	**12**	-1	-6
Y	-3	-4	-2	-4	0	-4	-4	-5	0	-2	-2	-4	-3	5	-5	-3	-3	-1	**8**	-3
V	0	-3	-2	-3	-2	-2	-2	-2	-2	3	1	-3	1	-2	-2	-1	0	-6	-3	**4**

$$R_{ij} = \frac{M_{ij}}{p_i}.$$

Whereas the probability matrix shown in Table 2.2 is not symmetrical, i.e. $M_{ij} \neq M_{ji}$, it holds that $R_{ij} = R_{ji}$. This is important, as the direction of a mutation is usually unknown when we try scoring a given pair of amino acids. Figure 2.7 shows the original amino acid background frequencies used in the construction of the original PAM matrices in addition to a set of background frequencies computed from a recent release of the protein database SwissProt. The amino acid frequencies in both data sets vary, ranging over almost an order of magnitude with tryptophan always being the rarest amino acid at 1%. The most frequent amino acid is glycine (8.9%) in the old data set and leucine (9.6%) in the more recent version. Perhaps surprisingly, the frequencies of tyrosine and phenylalanine have remained virtually unchanged.

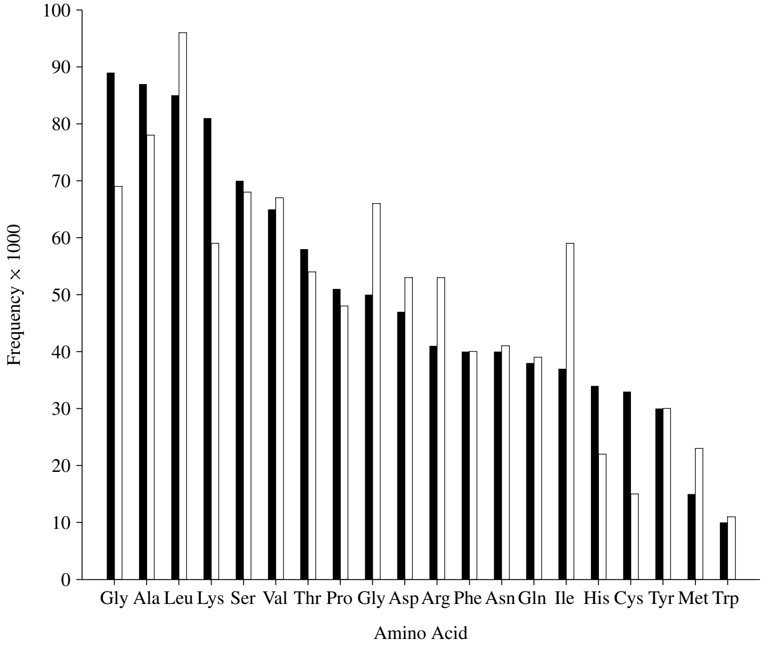


Fig. 2.7. Background frequencies of amino acids. The filled bars show the data used for the construction of the original PAM matrices in the 1970s [46]. The open bars display the amino acid frequencies calculated from the 172,233 protein sequences totaling 62,615,309 amino acids contained in the SwissProt database, release 46.2 of 2004.

The second step in converting transition probabilities into scores consists of taking the logarithm base 2 of R_{ij}:

$$s_{ij} = \log_2 (R_{ij}) .$$

The final entries, $\iota_{ij} = \text{round}(s_{ij})$, are known as log-odds scores.

PAM matrices were standard for aligning proteins until the early 1990s. They are still in use today, but have frequently been replaced by the BLOSUM system log-odds matrices [114].

2.4.2 BLOSUM Matrices

BLOSUM stands for **BLO**cks **SU**bstitution **M**atrices, because these matrices are derived from the BLOCKS database [113] consisting of blocks of ungapped protein alignments [113]. An example entry of the BLOCKS database is shown in Figure 2.8.

For deriving the entries in a BLOSUM matrix, consider an alignment of n sequences. We can form $\binom{n}{2}$ pairs of amino acids at each column of an alignment of n

```
ID    CLAUDIN1; BLOCK
AC    IPB003548C; distance from previous block=(3,4)
DE    Claudin-1 signature
BL    PR01377;  width=8; seqs=3; 99.5%=501; strength=972
CLD1_HUMAN|O95832   ( 152) MTPVNARY  83
CLD1_MOUSE|O88551   ( 152) LTPINARY 100

CLD1_RAT|P56745     ( 152) MTPVNARY  83
//
```

Fig. 2.8. Example entry in the BLOCKS database [202]. The sequences are taken from rat, mouse, and human claudin1. Claudins are a family of transmembrane proteins that form a major component of tight junctions [7, p.1068]. A BLOCKS entry consists of a four-line header and a sequence part. In the sequence part each line consists of the following elements: The SwissProt identifier, followed by the bracketed first position in the sequence included in the block, the actual amino acid sequence, and finally a sequence-weight. The latter is scaled such that 100 identifies the sequences that are most distant from all the other sequences contained in the block. Clusters are separated by a blank line.

sequences. Let a_{ij} be the number of pairs of amino acids i and j; then the observed frequency of this pair is

$$q_{ij} = \frac{a_{ij}}{l \cdot \binom{n}{2}};$$

where l is the length of the alignment. The frequency of this pair *expected* to occur by chance alone is

$$e_{ij} = \begin{cases} p_i^2; i = j \\ 2p_i p_j; i \neq j, \end{cases}$$

where p_i is the observed frequency of amino acid i. Entry s_{ij} in a BLOSUM matrix is then calculated as the log-odds of finding amino acids i and j at a homologous position in an alignment *vs.* the probability of observing this pair by chance alone:

$$s_{ij} = log_2 \left(\frac{q_{ij}}{e_{ij}} \right).$$

This is then multiplied by 2 and rounded to the nearest integer to yield the actual entries in the score matrix:

$$\iota_{ij} = \text{round}\,(2s_{ij}).$$

By way of an example, we now compute a BLOSUM matrix based on the single BLOCK shown in Figure 2.8. It consists of a total of 24 amino acids occurring at frequencies shown in Figure 2.9A. If for the time being we ignore clustering, we obtain the expected probabilities of finding pairs of amino acids shown in Figure 2.9B. Notice that the expectations of unobserved pairs of amino acids are set to zero, as they are ignored in subsequent steps of the derivation.

There is a total of 24 pairs of amino acids in the block and Figure 2.10 shows their raw counts and frequencies. By dividing entries in Figure 2.10B by the entries

A

i	Count	p_i
A	3	1/8
R	3	1/8
N	3	1/8
I	1	1/24
L	1	1/24
M	2	1/8
P	3	1/8
T	3	1/8
Y	3	1/8
V	2	1/12

B

	A	R	N	I	L	A	M	P	T	Y	V
A	1/64										
R	0	1/64									
N	0	0	1/64								
I	0	0	0	1/576							
L	0	0	0	0	1/576						
M	0	0	0	0	1/144	1/144					
P	0	0	0	0	0	0	1/64				
T	0	0	0	0	0	0	0	1/64			
Y	0	0	0	0	0	0	0	0	1/64		
V	0	0	0	1/144	0	0	0	0	0	1/144	

Fig. 2.9. The frequency of amino acid i, p_i, in the alignment block shown in Figure 2.8 without clustering. **A**: Amino acid counts and probabilities; **B**: probabilities of pairs of amino acids, e_{ij}; probabilities of unobserved pairs of amino acids, e.g. `RA`, are set to zero.

A

	A	R	N	I	L	A	M	P	T	Y	V
A	3										
R	0	3									
N	0	0	3								
I	0	0	0	0							
L	0	0	0	0	0						
M	0	0	0	0	2	1					
P	0	0	0	0	0	0	3				
T	0	0	0	0	0	0	0	3			
Y	0	0	0	0	0	0	0	0	3		
V	0	0	0	2	0	0	0	0	0	1	

B

	A	R	N	I	L	A	M	P	T	Y	V
A	1/8										
R	0	1/8									
N	0	0	1/8								
I	0	0	0	0							
L	0	0	0	0	0						
M	0	0	0	0	1/8	1/24					
P	0	0	0	0	0	0	1/8				
T	0	0	0	0	0	0	0	1/8			
Y	0	0	0	0	0	0	0	0	1/8		
V	0	0	0	1/8	0	0	0	0	0	1/24	

Fig. 2.10. Observed pairs of amino acids in the alignment block shown in Figure 2.8. **A**: Raw counts; **B**: frequencies.

in Figure 2.9B and taking the logarithm to the base 2, we get Figure 2.11A. Multiplication by 2 and rounding to the nearest integer gives us finally the substitution matrix in Figure 2.11B.

If we base our computations on the two clusters that make up the block in Figure 2.8, the expected probability of finding an amino acid is computed as the sum of the two probabilities of finding it in either cluster. For instance, the probability of finding a `M` is $1/2 \cdot 1/16 + 1/2 \cdot 1/8 = 3/32$ (Fig. 2.12A). As to the observed pairs of amino acids, we now make only comparisons between clusters. Hence, there are a total of 16 pairs as listed in Figure 2.12B.

A

	A	R	N	I	L	A	M	P	T	Y	V
A	3.0										
R	-	3.0									
N	-	-	3.0								
I	-	-	-	-							
L	-	-	-	-	-						
M	-	-	-	-	3.6	2.6					
P	-	-	-	-	-	-	3.0				
T	-	-	-	-	-	-	-	3.0			
Y	-	-	-	-	-	-	-	-	3.0		
V	-	-	-	3.6	-	-	-	-	-	2.6	

B

	A	R	N	I	L	A	M	P	T	Y	V
A	6										
R	-	6									
N	-	-	6								
I	-	-	-	-							
L	-	-	-	-	-						
M	-	-	-	-	7	5					
P	-	-	-	-	-	-	3				
T	-	-	-	-	-	-	-	3			
Y	-	-	-	-	-	-	-	-	3		
V	-	-	-	7	-	-	-	-	-	5	

Fig. 2.11. BLOSUM matrix computed from the alignment block shown in Figure 2.8. **A**: s_{ij}; **B**: t_{ij}. See text for details.

A

i	Count	p_i
A	3	1/8
R	3	1/8
N	3	1/8
I	1	1/32
L	1	1/32
M	2	3/32
P	3	1/8
T	3	1/8
Y	3	1/8
V	2	3/32

B

	A	R	N	I	L	A	M	P	T	Y	V
A	2										
R	0	2									
N	0	0	2								
I	0	0	0	0							
L	0	0	0	0	0						
M	0	0	0	0	1	1					
P	0	0	0	0	0	0	2				
T	0	0	0	0	0	0	0	2			
Y	0	0	0	0	0	0	0	0	2		
V	0	0	0	1	0	0	0	0	0	1	

Fig. 2.12. Clustered analysis of the alignment block shown in Figure 2.8. **A**: Count of amino acid i and its probability, p_i; **B**: Observed pairs of amino acids.

2.4.3 Comparison between PAM and BLOSUM

Recall that the PAM series of substitution matrices is constructed by extrapolating from a single starting matrix of substitution probabilities (Fig. 2.2). In contrast, BLOSUM matrices are based on frequencies of pairs of amino acids obtained from alignment blocks of various degrees of conservation. A BLOSUM62 matrix (Table 2.4), for instance, is derived from substitution probabilities based on alignments where all those sequences with identity $\geq 62\%$ are grouped and given the same weight as each of the ungrouped sequences. Hence, in contrast to PAM matrices, where a high PAM number corresponds to high sequence divergence, in the BLOSUM system high numbers correspond to a high degree of conservation and Table 2.5 shows examples of pairs of similar matrices from the BLOSUM and PAM series.

Having introduced substitution matrices, let us state more formally the definition of an alignment score. We consider an alphabet which contains the gap character.

Table 2.4. BLOSUM62 amino acid substitution matrix. Match scores are shown in bold.

	A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W	Y	V
A	**4**	-1	-2	-2	0	-1	-1	0	-2	-1	-1	-1	-1	-2	-1	1	0	-3	-2	0
R	-1	**5**	0	-2	-3	1	0	-2	0	-3	-2	2	-1	-3	-2	-1	-1	-3	-2	-3
N	-2	0	**6**	1	-3	0	0	0	1	-3	-3	0	-2	-3	-2	1	0	-4	-2	-3
D	-2	-2	1	**6**	-3	0	2	-1	-1	-3	-4	-1	-3	-3	-1	0	-1	-4	-3	-3
C	0	-3	-3	-3	**9**	-3	-4	-3	-3	-1	-1	-3	-1	-2	-3	-1	-1	-2	-2	-1
Q	-1	1	0	0	-3	**5**	2	-2	0	-3	-2	1	0	-3	-1	0	-1	-2	-1	-2
E	-1	0	0	2	-4	2	**5**	-2	0	-3	-3	1	-2	-3	-1	0	-1	-3	-2	-2
G	0	-2	0	-1	-3	-2	-2	**6**	-2	-4	-4	-2	-3	-3	-2	0	-2	-2	-3	-3
H	-2	0	1	-1	-3	0	0	-2	**8**	-3	-3	-1	-2	-1	-2	-1	-2	-2	2	-3
I	-1	-3	-3	-3	-1	-3	-3	-4	-3	**4**	2	-3	1	0	-3	-2	-1	-3	-1	3
L	-1	-2	-3	-4	-1	-2	-3	-4	-3	2	**4**	-2	2	0	-3	-2	-1	-2	-1	1
K	-1	2	0	-1	-3	1	1	-2	-1	-3	-2	**5**	-1	-3	-1	0	-1	-3	-2	-2
M	-1	-1	-2	-3	-1	0	-2	-3	-2	1	2	-1	**5**	0	-2	-1	-1	-1	-1	1
F	-2	-3	-3	-3	-2	-3	-3	-3	-1	0	0	-3	0	**6**	-4	-2	-2	1	3	-1
P	-1	-2	-2	-1	-3	-1	-1	-2	-2	-3	-3	-1	-2	-4	**7**	-1	-1	-4	-3	-2
S	1	-1	1	0	-1	0	0	0	-1	-2	-2	0	-1	-2	-1	**4**	1	-3	-2	-2
T	0	-1	0	-1	-1	-1	-1	-2	-2	-1	-1	-1	-1	-2	-1	1	**5**	-2	-2	0
W	-3	-3	-4	-4	-2	-2	-3	-2	-2	-3	-2	-3	-1	1	-4	-3	-2	**11**	2	-3
Y	-2	-2	-2	-3	-2	-1	-2	-3	2	-1	-1	-2	-1	3	-3	-2	-2	2	**7**	-1
V	0	-3	-3	-3	-1	-2	-2	-3	-3	3	1	-2	1	-1	-2	-2	0	-3	-1	**4**

Table 2.5. Examples of corresponding matrices from the BLOSUM and PAM series of amino acid substitution matrices [114].

BLOSUMn	PAMn
45	250
62	160
80	120

Then any two members x, y of this alphabet have a score $s(x, y)$ attached. Further, consider the sequences $S_1 = S_1[1]S_1[2]...S_1[m]$ and $S_2 = S_2[1]S_2[2]...S_2[n]$ and denote by S_1' and S_2' their gap-containing versions (cf. Figure 2.3). Then the score of an alignment of these two sequences of length l is simply the sum of the scores of pairs of characters:

$$\text{Score} = \sum_{i=1}^{l} s\left(S_1'[i], S_2'[i]\right).$$

Notice that this formula implies that each position in an alignment evolves independently. In many instances, for example in the case of protein coding sequences, this is an oversimplification, which nevertheless works remarkably well.

2.4.4 Application of Substitution Matrices

Consider an alignment of the polypeptides `AHY` and `GHF`, for example

```
AHY
GHF
```

According to the BLOSUM62 matrix (Table 2.4) the score of this alignment is $0+8+3=11$.

Substitution matrices such as BLOSUM62 are tuned to a specific evolutionary distance between the sequences being compared. Since over time any given position in a sequence is increasingly likely to have been hit by a mutation, PAM matrices with increasing PAM numbers become effectively more tolerant of non-identical amino acids and the same is true for BLOSUM matrices with decreasing BLOSUM numbers. Figure 2.13 shows this effect for optimal local alignments of the two *Hox* genes already shown in Figure 2.2. The alignment based on BLOSUM80 is sharply centered on the homeodomain. The alignment based on BLOSUM62 is extended by 24 amino acids at the C-terminus, while the N-terminus has remained unchanged. The alignment based on BLOSUM45, finally, has the same C-terminus as the BLOSUM62 alignment, but is extended by 386 amino acids at the N-terminus.

2.5 The Number of Possible Alignments

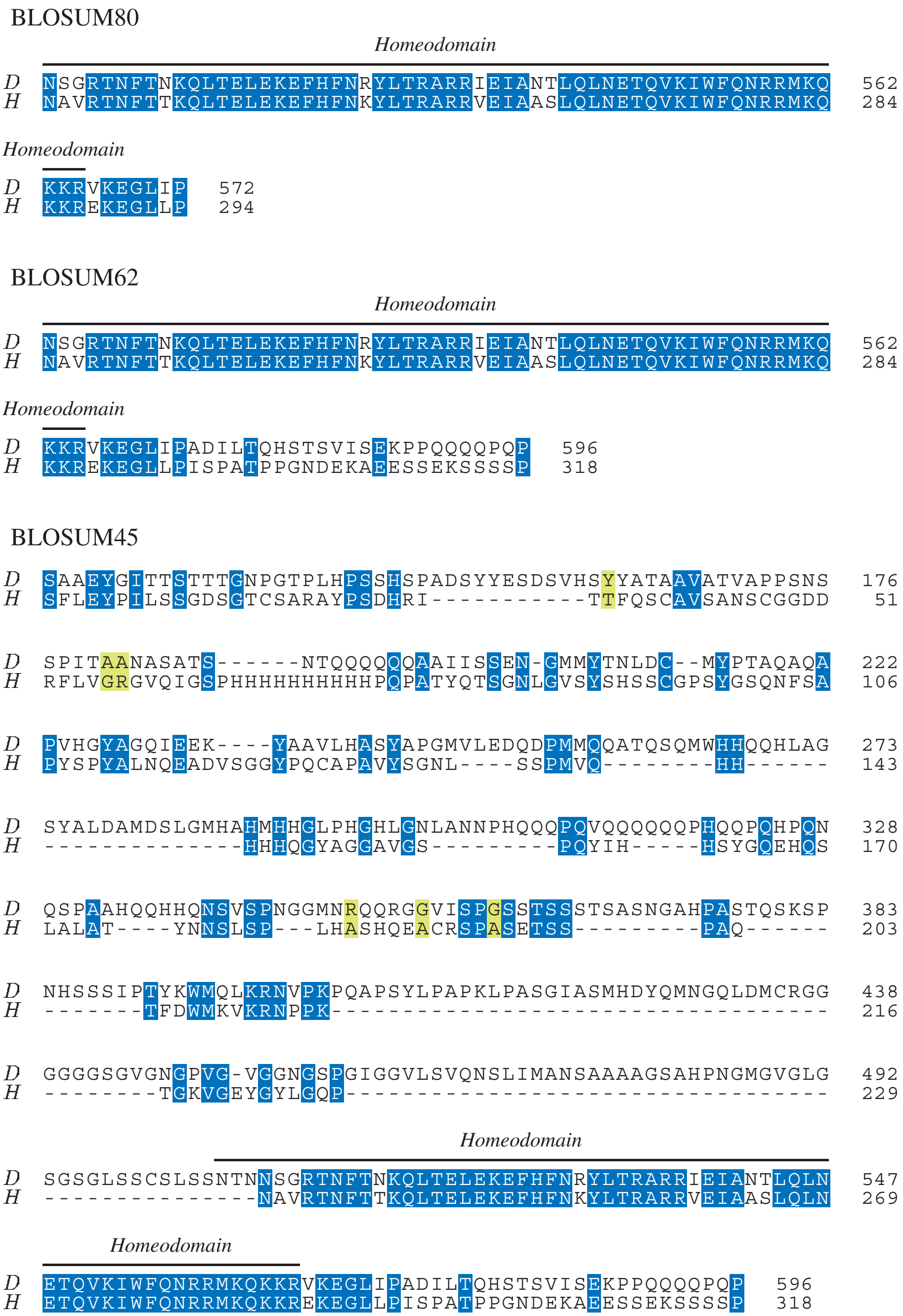
Now that we know how to score alignments the easiest way to decide our original question of which alignment is best might be to simply enumerate all possible alignments and look for the one with the highest score. Before embarking on such a "naïve" algorithm, let us estimate the number of possible alignments. As before, we consider two sequences $S_1 = S_1[1]S_1[2]...S_1[m]$ and $S_2 = S_2[1]S_2[2]...S_2[n]$, and further define $f(m,n)$ as the number of alignments that can be formed between them. The key insight necessary for calculating the number of possible alignments is that any alignment ends in one of exactly three ways:

$$\binom{S_1[m]}{-}, \binom{S_1[m]}{S_2[n]}, \text{or } \binom{-}{S_2[n]}.$$

Consider the effect of these three possible ends on the number of alignments that can be formed out of the remaining residues in the alignment. The ending $\binom{S_1[m]}{-}$ removes one residue from sequence S_1, $\binom{S_1[m]}{S_2[n]}$ removes one residue from sequences S_1 and S_2 each, and finally, the ending $\binom{-}{S_2[n]}$ removes one residue from sequence S_2. Therefore we can write the following recursion:

$$f(m,n) = f(m-1,n) + f(m-1,n-1) + f(m,n-1), \tag{2.2}$$

where the righthand side lists the number of alignments corresponding to each of the three types of ending. In addition to recursion (2.2) we need a "stop criterion", also known as boundary condition:



```
BLOSUM80
                              Homeodomain
D  NSGRTNFTNKQLTELEKEFHFNRYLTRARRIEIANTLQLNETQVKIWFQNRRMKQ   562
H  NAVRTNFTTKQLTELEKEFHFNKYLTRARRVEIAASLQLNETQVKIWFQNRRMKQ   284

Homeodomain
D  KKRVKEGLIP   572
H  KKREKEGLLP   294

BLOSUM62
                              Homeodomain
D  NSGRTNFTNKQLTELEKEFHFNRYLTRARRIEIANTLQLNETQVKIWFQNRRMKQ   562
H  NAVRTNFTTKQLTELEKEFHFNKYLTRARRVEIAASLQLNETQVKIWFQNRRMKQ   284

Homeodomain
D  KKRVKEGLIPADILTQHSTSVISEKPPQQQQPQP   596
H  KKREKEGLLPISPATPPGNDEKAEESSEKSSSSP   318

BLOSUM45
D  SAAEYGITTSTTTGNPGTPLHPSSHSPADSYYESDSVHSYYATAAVATVAPPSNS   176
H  SFLEYPILSSGDSGTCSARAYPSDHRI-----------TTFQSCAVSANSCGGDD    51

D  SPITAANASATS------NTQQQQQQAAIISSEN-GMMYTNLDC--MYPTAQAQA   222
H  RFLVGRGVQIGSPHHHHHHHHHHPQPATYQTSGNLGVSYSHSSCGPSYGSQNFSA   106

D  PVHGYAGQIEEK----YAAVLHASYAPGMVLEDQDPMMQQATQSQMWHHQQHLAG   273
H  PYSPYALNQEADVSGGYPQCAPAVYSGNL----SSPMVQ--------HH------   143

D  SYALDAMDSLGMHAHMHHGLPHGHLGNLANNPHQQQPQVQQQQQQPHQQPQHPQN   328
H  --------------HHHQGYAGGAVGS---------PQYIH-----HSYGQEHQS   170

D  QSPAAHQQHHQNSVSPNGGMNRQQRGGVISPGSSTSSSTSASNGAHPASTQSKSP   383
H  LALAT----YNNSLSP---LHASHQEACRSPASETSS---------PAQ------   203

D  NHSSSIPTYKWMQLKRNVPKPQAPSYLPAPKLPASGIASMHDYQMNGQLDMCRGG   438
H  -------TFDWMKVKRNPPK-----------------------------------   216

D  GGGGSGVGNGPVG-VGGNGSPGIGGVLSVQNSLIMANSAAAAGSAHPNGMGVGLG   492
H  --------TGKVGEYGYLGQP----------------------------------   229

                                     Homeodomain
D  SGSGLSSCSLSSNTNNSGRTNFTNKQLTELEKEFHFNRYLTRARRIEIANTLQLN   547
H  ---------------NAVRTNFTTKQLTELEKEFHFNKYLTRARRVEIAASLQLN   269

          Homeodomain
D  ETQVKIWFQNRRMKQKKRVKEGLIPADILTQHSTSVISEKPPQQQQPQP   596
H  ETQVKIWFQNRRMKQKKREKEGLLPISPATPPGNDEKAEESSEKSSSSP   318
```

Fig. 2.13. Optimal local alignments of two *Hox* proteins based on different substitution matrices. D: homeotic labial protein (*Lab*) from *D. melanogaster*; H: human *Hox-A1*; white on gray: identical amino acids; black on gray: similar amino acids.

$$f(0, m) = f(n, 0) = f(0, 0) = 1. \tag{2.3}$$

When trying to evaluate recursion (2.2), we have two options: (i) direct top down or (ii) indirect bottom up computation. We consider the top down approach first. It is called top down, because it starts from the desired end-result, say $f(2, 2)$, and then works its way "downwards" by repeated application of Equation (2.2), until the boundary condition (2.3) is reached. Figure 2.14 depicts this recursive process as a tree to be read from the top. Convince yourself from this graphic that the number of possible global alignments of two sequences of length 2 is 13 by summing all the terms in the tree corresponding to the boundary condition (2.3).

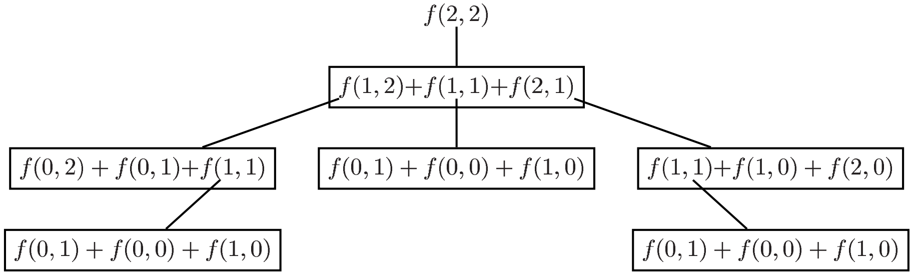
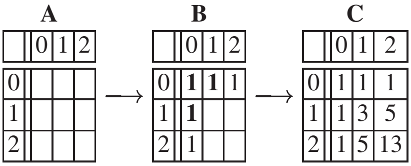


Fig. 2.14. Top down computation of the number of possible alignments for two sequences each of length 2 according to Equation (2.2).

Notice that $f(1, 1)$ appears three times in Figure 2.14, which means that this term is evaluated three times. In order to avoid such inefficient duplication of computational effort, we need to change our point of view. In the top down approach just discussed, the number of possibilities for the shortest alignments is computed last. If we compute them first and save these results as input for the computation of the next partial solution, the runtime of the algorithm becomes $O(m \cdot n)$ instead of exponential. To illustrate this, we start with an empty results matrix (Fig. 2.15A). This is initialized using boundary condition (2.3) by filling the first row and first column with 1's (Fig. 2.15B). The interpretation of this step is that there is only one way of aligning any sequence with a sequence of length zero, which consists exclusively of gaps. Now we can fill in the empty rows in our matrix by repeated application of recursion (2.2) going from left to right (or filling in columns of cells going from top to bottom). Hence, we quickly get the result that $f(2, 2) = 13$ (Fig. 2.15C). Notice that the results matrix is symmetrical. This corresponds to our intuition that the number of possible alignments does not depend on the way the two sequences concerned are labeled ($f(i, j) = f(j, i)$).

The idea used to speed up the computation of the number of possible alignments is to store subproblem solutions. When we augment this strategy by an optimization step, we arrive at the important computational method known as dynamic programming.



A

	0	1	2
0			
1			
2			

B

	0	1	2
0	**1**	**1**	1
1	**1**		
2	1		

C

	0	1	2
0	1	1	1
1	1	3	5
2	1	5	13

Fig. 2.15. Calculating the number of possible alignments from two sequences of length 2 by storing subproblem solutions. **A**: Empty solutions matrix; **B**: initialized solutions matrix—cells summed to compute the adjacent empty cell are shown in bold; **C**: filled in solutions matrix.

2.6 Global Alignment

In the subsequent sections three types of optimal pairwise alignment algorithms are dealt with: global, overlap, and local, which were first introduced at the beginning of this chapter (Fig. 2.1). All three types of alignment are computed using variations on the same underlying dynamic programming algorithm. Each algorithm is based on a two-dimensional table similar to those shown in Figure 2.15, which is known as a dynamic programming matrix. This data structure is subjected to the following three steps:

1. initializing of the dynamic programming matrix;
2. filling in the dynamic programming matrix with scores of optimal partial alignments;
3. extracting one or more alignments from the dynamic programming matrix; this is also called *traceback*.

We start by working through an example of global alignment, before stating the relevant rules more formally. This is followed by a briefer treatment of overlap and local alignments, which turn out to be simple extensions of the global alignment algorithm. Our example consists of the two sequences $S_1 = \texttt{ACCGTT}$ and $S_2 = \texttt{AGTTCA}$ throughout, which we wish to align using as scoring scheme $\mathrm{match} = 1$, $\mathrm{mismatch} = -1$, and $\mathrm{gap\ extension} = -1$. In other words, we use a linear gap model. The affine gap model is incorporated into the algorithm in Section 2.9.

Nomenclature

We need a certain amount of vocabulary when analyzing sequences. To a computer scientist the sequences of molecular biologists are *strings*. A string S is a list of characters ordered from left to right. The following notational conventions are used in this and subsequent chapters:

1. $|S|$ is the *length* of S.
2. $S[i..j]$ is a *substring* of S.
3. $S[1..i]$ is a *prefix* of S.
4. $S[i..|S|]$ is a *suffix* of S.

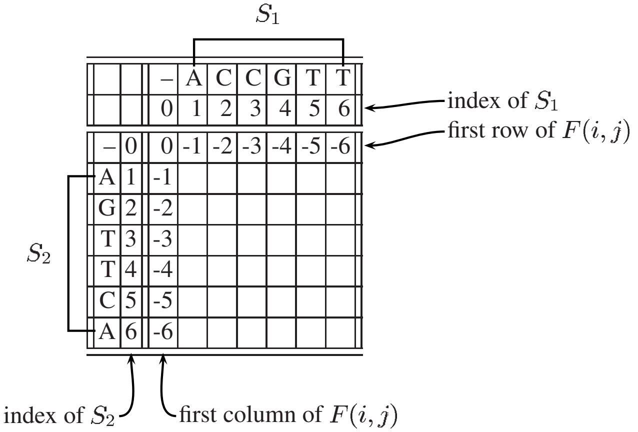
Characters in strings are labeled $1, \ldots, |S|$. As an example we consider

$$S = \texttt{ADDISABEBA}$$

In this case $|S| = 10$, `ADD` is a prefix, `IS` is a substring, and `BA` is a suffix. We will often use "string" and "sequence" interchangeably. Notice, however, that a *subsequence* is different from a substring. While a substring comprises only characters that are contiguous in the parent string, a subsequence may contain non-contiguous elements. For example, `DSB` is a subsequence, but not a substring of `ADDISABEBA`.

Initializing

In order to align our two example sequences, S_1 and S_2, we write each sequence preceded by a gap character along the two dimensions of the dynamic programming matrix shown in Figure 2.16, S_1 along the horizontal axis and S_2 along the vertical axis. We define the partial optimal score, $F(i, j)$, as the score of the optimal alignment of the prefixes $S_1[1, \ldots, i]$ and $S_2[1, \ldots, j]$. Since there is no alignment that consists of two aligned gaps, $F(0, 0) = 0$. To find $F(0, 1)$, we note that there is only a single way of writing the corresponding alignment, $\binom{A}{-}$, which has a score of -1. In general, there is only a single way of aligning a sequence exclusively with gaps and, hence, we can easily fill in the first row and also the first column of the dynamic programming matrix (Fig. 2.16).



		–	A	C	C	G	T	T
		0	1	2	3	4	5	6
–	0	0	-1	-2	-3	-4	-5	-6
A	1	-1						
G	2	-2						
T	3	-3						
T	4	-4						
C	5	-5						
A	6	-6						

Fig. 2.16. Initializing the dynamic programming matrix for global alignment.

Filling in the Dynamic Programming Matrix

In order to calculate the remaining values of $F(i, j)$, we need to consider the three ways in which an existing alignment can be extended by one unit. We have already encountered this in Section 2.5 when calculating the number of possible alignments: $\binom{S_1[m]}{-}$, $\binom{S_1[m]}{S_2[n]}$, $\binom{-}{S_2[n]}$. These three possibilities correspond to three moves in the dynamic programming matrix: horizontal for aligning a residue from S_1 with a gap, diagonal for aligning two residues, and vertical for aligning a residue from S_2 with a gap (Fig. 2.17). The aim is to find the maximum of the three possible values of $F(i,j)$ implied by the three possible extensions of an alignment. In this way the entire dynamic programming matrix is filled going from left to right and from top to bottom (Fig. 2.18). This takes the same amount of time as computing the number of possible alignments in Section 2.5: $O(m \cdot n)$.

		–	A	C	C	G	T	T
		0	1	2	3	4	5	6
–	0	0 ↘	-1 ↓	-2	-3	-4	-5	-6
A	1	-1 →						
G	2	-2						
T	3	-3						
T	4	-4						
C	5	-5						
A	6	-6						

Fig. 2.17. Dynamic programming matrix for global alignment: an empty cell surrounded by three filled cells can be evaluated by considering the three possible extensions of an alignment indicated by the three arrows and selecting the configuration with the maximum score. In the case shown, the three values to choose from are (i) $\rightarrow \equiv -1 + -1 = -2$, (ii) $\searrow \equiv 0 + 1 = 1$, and (iii) $\downarrow \equiv -1 + -1 = -2$. Hence the cell is filled with $\max(-2, 1, -2) = 1$ as shown in Figure 2.18.

		–	A	C	C	G	T	T
		0	1	2	3	4	5	6
–	0	0	-1	-2	-3	-4	-5	-6
A	1	-1	1	0	-1	-2	-3	-4
G	2	-2	0	0	-1	0	-1	-2
T	3	-3	-1	-1	-1	-1	1	0
T	4	-4	-2	-2	-2	-2	0	2
C	5	-5	-3	-1	-1	-2	-1	1
A	6	-6	-4	-2	-2	-2	-2	**0**

Fig. 2.18. Filled in dynamic programming matrix for global alignment. The score of the optimal alignment can be read from the bottom right cell; in this example it is 0.

The score in the bottom righthand corner of Figure 2.18 is the maximum score. In our example this happens to be 0, but it could be any number, positive or negative.

Notice how the repeated application of simple rules has yielded this result, even though there are 8,989 possible global alignments to choose from.

To summarize the strategy for filling in the global alignment matrix, we define the score of the optimal alignment between $S_1[1, ..., i]$ and $S_2[1, ..., j]$ as

$$F(i, j) = \max \sum_{i=1}^{l} s\left(S_1'[i], S_2'[i]\right).$$

The values of $F(i, j)$ are calculated through the recursions

$$\begin{aligned} F(i, 0) &= i \cdot g_{\mathrm{e}}, 0 \leq i \leq m, \\ F(0, j) &= j \cdot g_{\mathrm{e}}, 0 \leq j \leq n, \\ F(i, j) &= \max \begin{cases} F(i, j-1) + g_{\mathrm{e}} \\ F(i-1, j-1) + s(S_1[i], S_2[j]) \\ F(i-1, j) + g_{\mathrm{e}} \end{cases}, \end{aligned}$$

where g_{e} is the gap extension score.

Traceback

Having obtained the optimal score, we need to extract the corresponding alignment. This is usually done by a process referred to as “traceback”. At each step of the forward phase of the algorithm outlined above the cell in the dynamic programming matrix from which the current value $F(i, j)$ has been derived is “remembered” through a back pointer. We visualize this as a little arrow that points to the cell from which the entry in the current cell was computed (Fig. 2.19). There might be up to three back pointers per cell corresponding to three cooptimal alignment extensions.

Once the dynamic programming matrix has been filled in, the back pointers are followed from the bottom righthand corner to the top lefthand corner and the corresponding alignment is constructed in reverse order (Fig. 2.19).

There is only one possible traceback path in Figure 2.19, which means that the corresponding global alignment is unique. This is not necessarily the case and if more than one traceback path is possible, there is no way to distinguish the resulting cooptimal alignments under the model. In practice, most implementations ignore this problem and return an arbitrary member of the potentially large set of cooptimal global alignments.

Next we consider overlap alignment (cf. Fig. 2.1), which is applied in the context of a widely used sequencing strategy known as shotgun sequencing.

2.7 Shotgun Sequencing and Overlap Alignment

There are two possibilities for sequencing large pieces of DNA: (i) “Walk” along the target sequence in steps of a couple of hundred nucleotides. This is slow as a new primer needs to be synthesized for every step in the walk. Moreover, walking can

		–	A	C	C	G	T	T
		0	1	2	3	4	5	6
–	0	**0**	← -1	← -2	← -3	← -4	← -5	← -6
A	1	↑ -1	↖ **1**	← **0**	← **-1**	← -2	← -3	← -4
G	2	↑ -2	↑ 0	↖ 0	↖← -1	↖ **0**	← -1	← -2
T	3	↑ -3	↑ -1	↖↑ -1	↖ -1	↑ -1	↖ **1**	↖← 0
T	4	↑ -4	↑ -2	↖↑ -2	↖↑ -2	↖↑ -2	↖↑ 0	↖ **2**
C	5	↑ -5	↖↑ -3	↑ -1	↑ -1	← -2	↑ -1	↑ **1**
A	6	↑ -6	↑ -4	↖ -2	↖ -2	← -2	↑ -2	↑ **0**

```
ACCGTT--
A--GTTCA
```

Fig. 2.19. Dynamic programming matrix with back pointers. The optimal global alignment (*bottom*) corresponds to the path marked in bold (*top*).

only be carried out sequentially, as the primer sequence necessary for a given step is unknown until the previous step has been taken. An alternative approach is (ii) shotgun sequencing, where random fragments of the target sequence are cloned into a vector and sequenced using always the same primers complementary to the vector.

Shotgun sequencing [215] is motivated by the fact that each DNA sequencing reaction identifies only between 400 and 800 nucleotides, a very small portion of even small genomes (Table 2.6). With the shotgun approach many sequencing reactions can be carried out in parallel. The resulting rapidly generated fragments overlap and we will explain how this fact can be used to assemble the target sequence. But before that we ask, how many such fragments are necessary for successful assembly?

Let L be the length of the target sequence. When sequencing random bases from that sequence, the probability of not sequencing a specific nucleotide is

$$P_0 = \left(1 - \frac{1}{L}\right)^s \approx e^{-\frac{s}{L}},$$

where s is the number of bases sequenced. The complementary probability that a base has been sequenced is

$$P = 1 - e^{-c},$$

where $c = s/L$ is called the *coverage*. Hence, with six-fold coverage of a genome, 99.8% of nucleotides are included in the data set ready for computational assembly.

The idea underlying the assembly of shotgun fragments is known as *overlap alignment*, sometimes also referred to as *end space free alignment* [99, ch. 11]. With this approach, gaps at either ends of the alignment are not penalized (Fig. 2.1). This is done by initializing the first row and column of the dynamic programming matrix to zero and then filling the matrix exactly as for the global alignment (Fig. 2.20). For a linear gap model the corresponding recursions are

$$
\begin{aligned}
F(i,0) &= 0, \\
F(0,j) &= 0, \\
F(i,j) &= \max \begin{cases} F(i,j-1) + g_{\mathrm{e}} \\ F(i-1,j-1) + s(S_1[i],S_2[j]) \\ F(i-1,j) + g_{\mathrm{e}} \end{cases} .
\end{aligned}
$$

The traceback step starts from the maximum score in the last row or column and the positions left unaccounted for are filled with gaps. Notice that the cell containing the entry -1 on the traceback path in Figure 2.20 contains three back pointers, implying that there are three cooptimal overlap alignments of the two input sequences. Moreover, in contrast to global alignments, there might be more than one cooptimal starting point for the traceback.

Table 2.6. Milestones of genome sequencing; bp: base pair, Mb: 10^6 base pairs, Gb: 10^9 base pairs.

Year	Organism	Genome Size	Commentary
1978	bacteriophage ϕX174	5386 bp	first genome [214]
1982	bacteriophage λ	48.502 bp	shotgun sequencing [215]
1995	*Haemophilus influenzae*	1.8 Mb	first free-living organism [78]
1996	*Saccharomyces cerevisiae*	12 Mb	first eucaryote [88]
1998	*Caenorhabditis elegans*	97 Mb	first metazoan [243]
2000	*Arabidopsis thaliana*	120 Mb	first plant [245]
2000	*Drosophila melanogaster*	180 Mb	first metazoan with whole-genome shotgun [3]
2001/2003	*Homo sapiens*	3.1 Gb	aim of the human genome project [128, 252, 39]
2002	*Mus musculus*	2.5 Gb	comparative pharmacogenomics [40]
2005	*Pan troglodytes*	3.1 Gb	closest relative of humans [223]

2.8 Local Alignment

Two proteins often share only homologous domains rather than homology across their entire lengths. In this case we seek a local, rather than a global alignment as illustrated in Figure 2.1. Put slightly more formally, the local alignment problem is solved by finding a maximally scoring pair of substrings. This requirement increases the space of possible alignments, since all pairs of substrings need to be considered. The sequence $S = \texttt{ACT}$, for example, contains the substrings $\{\texttt{A},\texttt{C},\texttt{T},\texttt{AC},\texttt{CT},\texttt{ACT}\}$. In general, a sequence of length m contains $\binom{m+1}{2}$ substrings. For two sequences of lengths m and n there are correspondingly $\binom{m+1}{2}\binom{n+1}{2}$ pairs of substrings to be compared. Naïvely, we would need to compute a global alignment for each one of these and then look for the highest scoring pair. However, a slight variation on the global alignment scheme ensures that the time requirement for local alignments is equal to that for global alignments, i.e. $O(m \cdot n)$. This variation consists in taking the maximum over the three possible extensions of an existing alignment and zero (Fig. 2.21). The corresponding recursions are

		–	A	C	C	G	T	T
		0	1	2	3	4	5	6
–	0	**0**	← **0**	← **0**	← 0	← 0	← 0	← 0
A	1	↑ 0	↖ 1	← 0	↖←↑ **-1**	↖↑ -1	↖↑ -1	↖↑ -1
G	2	↑ 0	↑ 0	↖ 0	↖← -1	↖ **0**	← -1	↖←↑ -2
T	3	↑ 0	↖←↑ -1	↖↑ -1	↖ -1	↑ -1	↖ **1**	↖← 0
T	4	↑ 0	↖← -1	↖←↑ -2	↖↑ -2	↖↑ -2	↖↑ 0	↖ **2**
C	5	↑ 0	↖← -1	↖ 0	↖← -1	← -2	↑ -1	↑ 1
A	6	↑ 0	↖ 1	← 0	↖← -1	↖← -2	↑ -2	↑ 0

```
ACCGTT--
--AGTTCA
```

Fig. 2.20. Dynamic programming matrix for overlap alignment (*top*) and one of the corresponding three cooptimal alignments (*bottom*). The traceback path corresponding to the alignment is marked in bold. It starts from the maximum score in the bottom row or last column.

$$
\begin{aligned}
F(i,0) &= 0,\\
F(0,j) &= 0,\\
F(i,j) &= \max\begin{cases} F(i,j-1)+g_{\mathrm{e}}\\ F(i-1,j-1)+s(S_1[i],S_2[j])\\ F(i-1,j)+g_{\mathrm{e}}\\ 0 \end{cases}.
\end{aligned}
$$

The traceback step starts from the maximum score in the entire dynamic programming matrix and proceeds until a cell with score zero is reached. As noted already for overlap alignments, there might be more than one possible starting point for a local alignment, and more than one cooptimal alignment may correspond to each starting point. Also, in the case of local alignments it is often interesting to include high-scoring but suboptimal alignments in the result set. These might correspond to weaker but still significant homologies.

2.9 Accommodating Affine Gap Costs

So far we have simplified matters by using a linear gap model. However, as explained in Section 2.3, the affine gap scoring scheme is more realistic by distinguishing between gap opening and gap extension. The incorporation of the affine gap model in the dynamic programming algorithms seen so far follows from the insight that a gap insertion constitutes either the extension of an existing gap, or the opening of a new gap and its concomitant extension by one. To account for this, the three types of alignment endings need to be considered again. Let $H(i,j)$ denote the maximum score of an alignment ending in $\binom{S_1[m}{-}$, $F(i,j)$ the maximum score of an alignment

		–	A	C	C	G	T	T
		0	1	2	3	4	5	6
–	0	0	0	0	0	0	0	0
A	1	0	↖ 1	0	**0**	0	0	0
G	2	0	0	0	0	↖ **1**	0	0
T	3	0	0	0	0	0	↖ **2**	↖← 1
T	4	0	0	0	0	0	↖↑ 1	↖ **3**
C	5	0	0	↖ 1	↖ 1	0	0	↑ 2
A	6	0	↖ 1	0	0	0	0	↑ 1

```
GTT
GTT
```

Fig. 2.21. Dynamic programming matrix for local alignment (*top*). The path corresponding to the optimal alignment (*bottom*) is marked in bold. In contrast to global and overlap alignments (Figs. 2.19 and 2.20), the local alignment consists of a pair of substrings from the input sequences.

ending in $\binom{S_1[m]}{S_2[n]}$, and $V(i,j)$ the maximum score of an alignment ending in $\binom{-}{S_2[n]}$. In order to calculate the recursions, we note not only $F(i,j)$, as we have done so far, but also $H(i,j)$ and $V(i,j)$. Then the following recursions can be applied to the global alignment problem:

$$
\begin{aligned}
F(0,0) &= 0,\\
F(i,0) &= V(i,0) = -\infty; i > 0,\\
F(0,j) &= H(0,j) = -\infty; j > 0,\\
H(i,0) &= g_o + i \cdot g_e,\\
V(0,j) &= g_o + j \cdot g_e,\\
F(i,j) &= \max \begin{cases} F(i-1,j-1) + s(S_1[i],S_2[j]) \\ H(i-1,j-1) + g_o + g_e \\ V(i-1,j-1) + g_o + g_e \end{cases},\\
H(i,j) &= \max \begin{cases} F(i,j-1) + g_o + g_e \\ H(i,j-1) + g_e \end{cases},\\
V(i,j) &= \max \begin{cases} F(i-1,j) + g_o + g_e \\ V(i-1,j) + g_e \end{cases}.
\end{aligned}
$$

An example application of this strategy is shown in Figure 2.22 where the score scheme $g_o = -5$, $g_e = -2$, $\mathrm{match} = 1$, and $\mathrm{mismatch} = -3$ is applied to the two sequences $S_1 =$ `ATC` and $S_2 =$ `AC`. The traceback starts in the bottom righthand corner of the matrix with the highest entry there; in our case this is matrix F with the entry -5. From there it moves diagonally, i.e. the nascent alignment consists of $\binom{\mathtt{C}}{\mathtt{C}}$. At the same time, the traceback switches from matrix F to matrix V, which indicates that a gap is opened and extended by one position in the sequence written vertically, yielding $\binom{\texttt{TC}}{\texttt{-C}}$. The next move switches back to F, thereby closing the gap, and adds one further position to return the final alignment, $\binom{\texttt{ATC}}{\texttt{A-C}}$. The score of this is -5, as demanded by the affine gap model defined in Equation (2.1).

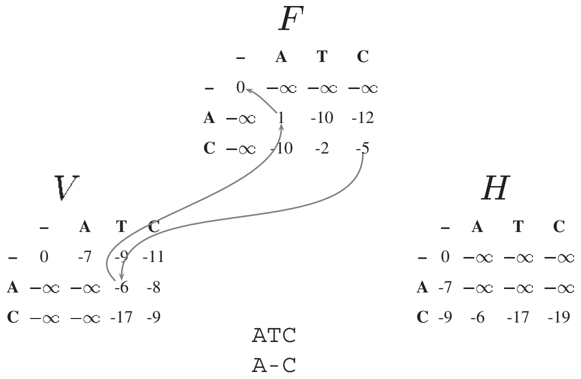


Fig. 2.22. The three matrices needed for efficiently computing alignments with affine gap costs. Only the traceback for the final alignment (underneath the matrices) is shown (*arrows*). See text for further explanations.

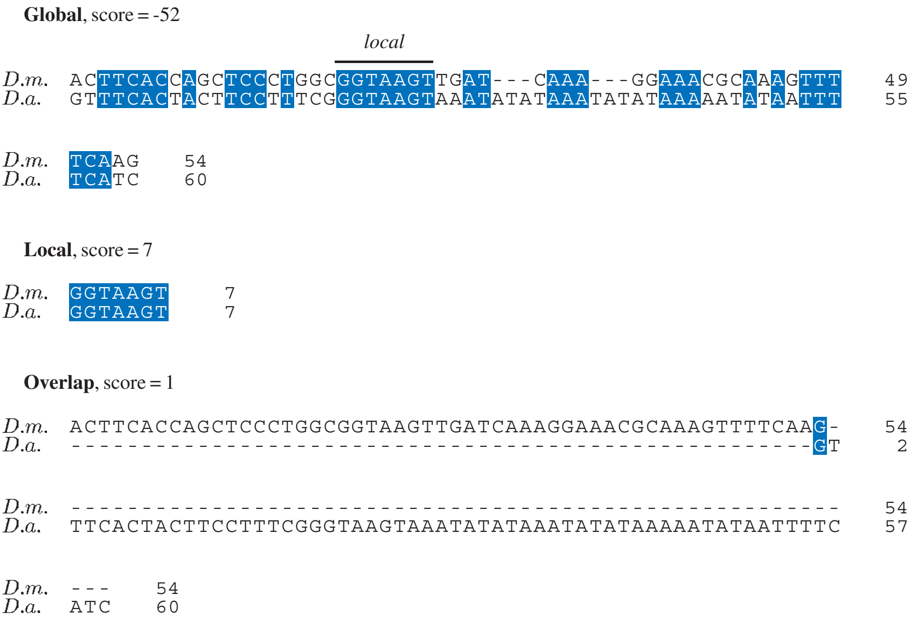
Overlap and local alignment strategies can be adapted analogously for dealing with affine gap costs.

2.10 Maximizing *vs.* Minimizing Scores

We noted in Section 2.2 that there are two ways of interpreting alignments. The first is that they *minimize* the implied number of evolutionary changes. Alternatively, one may say that they *maximize* the implied amount of sequence conservation. At a first glance these two interpretations appear to be equivalent. This begs the question, why all the scoring schemes we have covered in this chapter are of the maximizing kind? The reason for this is that local alignments depend on the cutoff score of zero, which implies as hidden assumption that the expected score of a random pair of residues is negative. If this condition were not met, the score of a local alignment would grow through the addition of random pairs of residues. With a minimizing scoring scheme no such natural numerical barrier exists. In order to make global alignment consistent with the local algorithm, global alignments are also usually calculated by maximizing a score even though in this case a minimizing scheme would work just as well because it always extends over the entire lengths of the input sequences.

2.11 Example Application of Global, Local, and Overlap Alignment

Figure 2.23 shows the result of applying the global, local, and overlap alignment algorithms to the same pair of short DNA sequences. The global alignment looks reasonable, indicating that the sequences are indeed homologous across their entire lengths. The local alignment picks the block marked *local* in the global alignment. This is the highest scoring pair of substrings from the two sequences. In contrast to the global and local alignments, the overlap alignment has no biological meaning; there simply is no substantial overlap at the end of the sequences to be detected.



Global, score = -52

```
                                    local
D.m.  ACTTCACCAGCTCCCTGGCGGTAAGTTGAT---CAAA---GGAAACGCAAAGTTT    49
D.a.  GTTTCACTACTTCCTTTCGGGTAAGTAAATATATAAATATATAAAAATATAATTT    55

D.m.  TCAAG    54
D.a.  TCATC    60
```

Local, score = 7

```
D.m.  GGTAAGT    7
D.a.  GGTAAGT    7
```

Overlap, score = 1

```
D.m.  ACTTCACCAGCTCCCTGGCGGTAAGTTGATCAAAGGAAACGCAAAGTTTTCAAG-    54
D.a.  -----------------------------------------------------GT     2

D.m.  -------------------------------------------------------    54
D.a.  TTCACTACTTCCTTTCGGGTAAGTAAATATATAAATATATAAAAATATAATTTTC    57

D.m.  ---    54
D.a.  ATC    60
```

Fig. 2.23. Optimal alignments computed using the three types of alignment algorithm discussed in this chapter. The input sequences consist of 60 nucleotides taken from the *Adh* locus in two species of *Drosophila*. *D.m.*: *D. melanogaster*; *D.a.*: *D. adiastola*. The scoring scheme was: match = 1, mismatch = −3, gap opening = −5, gap extension = −2.

2.12 Summary

Alignments are used for sequence comparison and typically model two evolutionary events: point mutations and indels. A pairwise alignment between two sequences is generated by writing the two sequences on top of each other such that either two residues or a residue and a gap symbol are paired. An *optimal* alignment is one that either minimizes the implied number of evolutionary changes or maximizes a particular scoring function. Residues are scored differently from indels. Match and mismatch scores are applied to nucleotides, while PAM and BLOSUM amino acid substitution matrices are used for protein sequences. Gaps are usually modeled using the affine score function, which distinguishes between gap opening and gap extension. Since the scores of alignments can be defined recursively, dynamic programming is used for efficiently calculating optimal alignments. There are three major types of such alignments, global, local, and overlap. A global alignment is based on the assumption that the sequences are homologous over their entire lengths. In contrast, local alignments search for local homologies. Finally, overlap alignments are used for the assembly of overlapping sequence fragments, as might be generated in the course of a shotgun sequencing project.

2.13 Further Reading

An advanced treatment of alignment algorithms is given in the textbook by Waterman [253], who is also one of the coinventors of the optimal local alignment algorithm and a number of other algorithms in computational biology.

2.14 Exercises and Software Demonstrations

2.1. Consider the following DNA alignment:

```
AACTCA---G
ATCTGATTTG
```

What is its score under the following scoring scheme: $\mathrm{match} = 1$, $\mathrm{mismatch} = -3$, $\mathrm{gap\ opening} = -5$, $\mathrm{gap\ extension} = -2$?

2.2. What is the probability of drawing a pair of distinct nucleotides from a nucleotide database under the assumption that all residues have the same frequency?

2.3. The genome of *Mycoplasma genitalium* has the following composition:

Nucleotide	Frequency
A	0.346
C	0.158
G	0.159
T	0.337

What is the probability of drawing a pair of distinct nucleotides from the genome of *M. genitalium*?

2.4. Consider the following protein alignment:

```
AAWGCCA
AVWGCCL
```

What is its score using the BLOSUM62 substitution matrix (Table 2.4)?

2.5. Use the `bioinformer` program `Alignment→Protein Substitution Matrices` (Section A.1.1) to calculate the expected difference between a pair of protein sequences separated by 5 PAM and by 50 PAM. Do these values differ from the PAM number? Explain.

2.6. Imagine you are given the task of replacing the old system of match and mismatch scores for DNA sequence alignments by PAM matrices. From sequence alignments you have already computed the following mutation probability matrix for sequences that have diverged by one PAM.

	A	C	G	T
A	0.97	0.01	0.01	0.01
C	0.01	0.97	0.01	0.01
G	0.01	0.01	0.97	0.01
T	0.01	0.01	0.01	0.97

In addition, the background frequencies of all nucleotides are $1/4$.

1. Compute the PAM1 substitution matrix.
2. Compute the PAM2 substitution matrix.

2.7. Use the information given in Figure 2.12 to compute the final BLOSUM score matrix for the clustered BLOCKS data.

2.8. Consider two random nucleotide sequences of equally likely nucleotides and a scoring scheme of $\mathrm{match} = 3$, $\mathrm{mismatch} = -5$. What is the expected score of a random pair of residues? Can this scoring scheme be applied in the context of local alignment algorithms?

2.9. What is the number of global alignments between two sequences of lengths 3 and 4?

2.10. List all substrings of sequence $S =$ ACG.

2.11. What is the number of pairs of substrings that can be formed between $S_1 =$ ACG and $S_2 =$ ACGT?

2.12. What is the number of local alignments between $S_1 =$ ACG and $S_2 =$ ACGT?

2.13. Imagine a molecular biologist sends you a file containing the shotgun sequencing data for *Escherichia coli* strain K12. The genome of this bacterium consists of 4,639,675 nucleotides and the shotgun sequencing data consists of 82,500 fragments of an average length of 500 bp.

1. What is the coverage of this genome?

2. How many nucleotides do you expect to have remained unsequenced?
3. What coverage would have been necessary to obtain an expectation of a single unsequenced nucleotide?

2.14. Write a computer program for calculating the number of global alignments that can be formed between two sequences. How many different global alignments can be formed between two sequences of lengths 390 and 400? Compare your answer to that given by the `bioinformer` program `Number of Alignments` (Section A.1.2).

2.15. Write down the missing two cooptimal alignments implied by the dynamic programming matrix in Figure 2.20.

2.16. Construct an algorithm for computing the number of cooptimal global alignments.

2.17. Compute the optimal global alignment between S_1 = `ATA` and S_2 = `ATTA` using dynamic programming and a scoring scheme of $\mathrm{match} = 1$, $\mathrm{mismatch} = -1$, $\mathrm{gap\ extension} = -1$, $\mathrm{gap\ opening} = 0$. What is the score of the optimal alignment? What is the optimal alignment? Is the optimal alignment unique? Compare your results to those of the `bioinformer` program `Pairwise Alignment` (Section A.1.3).

2.18. Global alignment can be regarded as a special case of local alignment. Rewrite the recursions for local alignment such that they also become applicable to the global case.

3

Biological Sequences and the Exact String Matching Problem

> *One needs some form of memory with which any required entry can be reached at short notice. This difficulty presumably used to worry the Egyptians when their books were written on papyrus scrolls. It must have been slow work looking up references in them, and the present arrangement of written matter in books which can be opened at any point is greatly to be preferred.*
>
> *Alan M. Turing [117, p. 319]*

The human genome consists of $3.1 \cdot 10^9$ nucleotides [128, 252]. A printout of this text would generate a rather massive book, given that Shakespeare's collected works amount to $4.2 \cdot 10^6$ characters—blanks, punctuations, and all. In other words, a printout of the human genome would result in a small library of over 730 volumes, each the size of Shakespeare's collected works. If we wanted to look up the usage of, say, the word "phoenix" in Shakespeare, we'd need to leaf through his oeuvre from start to finish. That is, such a search would take time proportional to the size of the bard's works. However, if we had access to a concordance, i.e. an index, to the works of Shakespeare, we could find the occurrences of "phoenix" much more rapidly. In this case, our search would take time proportional to the length of the word or phrase we are looking for. This old but nevertheless very pleasing insight should not let us forget that the construction of a concordance is a painstaking business (the concordance to Shakespeare's works by itself comprises nine volumes). Hence there is a trade-off between the effort of constructing a concordance and the total time saved through its usage.

In this chapter we shall encounter computational methods that emulate both types of search mentioned so far: (i) linear in the size of the text and (ii) independent of the size of the text through indexing.

3.1 Exact *vs.* Inexact String Matching

Biologists investigating a nucleotide or amino acid sequence often need to look up specific patterns in this sequence. A simple example would be to search for potential start codons in a protein-coding mRNA sequence. In this case we would be looking for all the positions where an *exact* match between a pattern (`ATG`) and a text (the mRNA sequence) starts.

Alternatively, many protein families are characterized by *inexact* matches to signature sequences. PROSITE is a data base dedicated to the collection of such signature sequences [226]. For example, mitogen-activated protein kinases (MAP kinases) form a class of threonine/serine kinases that are involved in many central cellular signal transduction pathways. They are characterized by the PROSITE-signature

`F-x(10)-R-E-x(72,86)-R-D-x-K-x(9)-C.`

Hyphens separate the elements of the pattern, letters refer to amino acids, and an `x` indicates any amino acid. Bracketed numbers denote the repeat length of a residue; if this repeat length varies, the range of this variation is quoted in the brackets. The pattern reproduced above might appear somewhat long-winded as it is spread out over approximately 100 residues. But it has the very useful property that all MAP kinases conform to it, while no other proteins in the comprehensive protein sequence database SwissProt [25] match this pattern.

A PROSITE-signature is a compact description of a potentially large set of sequences. Strings that describe sets of strings through a special syntax are known as *regular expressions* and PROSITE-signatures are a particular kind of regular expression [80]. Notice that in order to generate such a signature, the approximate positions of conserved regions have to be known. These can only be discovered by comparing homologous proteins, i.e. proteins with a shared recent common ancestor. The detection of such homologous proteins is usually carried out with a different kind of inexact matching known as *alignment*. Alignments are dealt with in Chapters 2, 4, and 5. In this chapter we look at exact matching and matching with at most k mismatches. We will see that exactly matching a pattern with a text can be achieved in time proportional to the length of the text, n, using a variety of classical methods. In the first part of this chapter we introduce an example algorithm that achieves this time bound. Next we show how we can search for a *set* of patterns of total length m rapidly in $O(m + n + o)$ time, where o is the number of occurrences of the patterns. Perhaps surprisingly, it is possible to improve dramatically on these strategies and to execute exact pattern searches in time proportional to the length of the pattern through the use of an index. We start our survey of exact string matching methods with a simple, "naïve" approach to clarify the issues involved.

3.2 Naïve Pattern Matching

When comparing two characters, their identity is called a *match*, the converse a *mismatch*. We usually talk about two kinds of strings, the text, T, with length $|T| = n$, and the pattern, P, with length $|P| = m$, where $m \leq n$ and usually $m \ll n$. Solving the *exact string matching problem* then amounts to finding all occurrences of P in T. If we take as an example the DNA sequences $T =$ `CCACAGACACAT` and $P =$ `ACA`, then P occurs three times in T starting at positions 3, 7, and 9.

A *naïve method* for solving the exact matching problem is to align the left ends of P and T and to compare each character going from left to right. If a mismatch is found, or all letters in P have been matched, P is shifted to the right by one position

```
        Phase 1      Phase 2      Phase 3
Text    CCACAGACACAT CCACAGACACAT CCACAGACACAT
Pattern ACA            ACA            ACA
        ^              ^              ***
```

Fig. 3.1. First three phases in naïve algorithm for exact string matching. The algorithm proceeds from left to right; mismatches detected by the algorithm are indicated by a circumflex (ˆ), matches by a star ($\star$).

(Fig. 3.1). In the worst case P and T consist of a single type of character. Then the naïve algorithm performs $m(n - m + 1)$ comparisons and its runtime is therefore $O(m \cdot n)$. Through *preprocessing* of P and T (as we will explain in Section 3.3) the exact matching problem can be solved much more effectively in $O(m+n)$ time, that is, in time linear in the size of the text.

3.3 String Searching in Linear Time

There is a rich tradition of string matching algorithms in the computer science literature [99]. Here we describe one algorithm that runs in time linear in the length of the text, the Z-algorithm [99, p. 8ff].

The fundamental concept of the algorithm is that of a *Z-value*. Given a string S and a string position i, the Z-value $Z_i(S)$ is the length of the longest substring of S that starts at i and matches a prefix of S. If we take as an example string the DNA sequence $S = \texttt{CACAG}$, then $Z_2 = 0$, $Z_3 = 2$, $Z_4 = 0$, and so on. Notice that Z_1 will always be equal to $|S|$ and is omitted.

As an extension of the concept of Z-values, the substring $[i..i + Z_i - 1]$ is called a *Z-box* at every position i where $Z_i > 0$. In order to simplify our notation for the algorithm, let r and l denote the righthand and lefthand borders of the Z-box corresponding to the current position in the text.

The algorithm to determine Z-values for string S, $|S| = m$, starts by comparing the prefix of $S[2..m]$ with the prefix of S until a mismatch is found. The length of the matching substring is Z_2. If $Z_2 > 0$, we set r to $Z_2 + 1$ and l to 2. Assume now that we already know Z_i for $i = 2, \ldots, k - 1$ and the borders of the corresponding Z-box, r and l. In order to compute Z_k we use the Z-values calculated previously:

1. If $k > r$, find new Z-box. If $Z_k > 0$, $r \leftarrow k + Z_k - 1$ and $l \leftarrow k$.
2. If $k \leq r$, we define $k' = k - l + 1$ and $\beta = S[k..r]$ and consider two cases:
   a) If $Z_{k'} < |\beta|$, $Z_k \leftarrow Z_{k'}$, r and l remain unchanged.
   b) If $Z_{k'} \geq |\beta|$, compare characters starting at positions $r + 1$ and $|\beta| + 1$, until a mismatch at position q is found; then $Z_k \leftarrow q - k$, $r \leftarrow q - 1$, and $l \leftarrow k$.

To solve the exact string matching problem, we consider the string $S = P\$T$, where \$ is a character that occurs neither in T nor in P. Every time $Z_i = |P|$, an occurrence of P in T has been found. Z-values can be computed in $O(m + n) \approx$

$O(n)$ and hence the exact string matching problem has been solved in time linear in the size of the text.

Let us work through a concrete application of this algorithm. Our input string is again $S = \mathtt{CACAG}$, and we search for the pattern $P = \mathtt{ACA}$. Figure 3.2 shows the corresponding setup with the Z-values computed so far. We now wish to compute Z_7. Currently $l = 6$ and $r = 8$; since $k = 7 \leq r$, we compute $k' = 7 - 6 + 1 = 2$, and $\beta = S[7..8]$ as marked in Figure 3.2. Since $Z_2 = 0 \leq |\beta| = 2$, case 2(a) applies and hence $Z_7 = 0$.

index 123456789
Z_i 901003?
String $\mathtt{ACA\$CA}\underbrace{\mathtt{CA}}_{\beta}\mathtt{G}$

Fig. 3.2. Z-algorithm: searching for `ACA` in `CACAG`.

The Z-algorithm deals with a single pattern at a time. However, it is also possible to simultaneously match an entire set of patterns against a text. The corresponding algorithms are based on trees, which are introduced next.

3.4 Trees

The string search methods explained in the subsequent sections make use of a data structure that reappears in many parts of this book: a tree. Trees are used as index structures for rapid string searching in this chapter, as guide trees for multiple sequence alignment in Section 5.3, as phylogenies in Chapter 8, and as coalescent trees for simulating gene samples in Chapter 11 and again in Section 12.3. In fact, this book itself has a hierarchical, that is, tree structure.

We follow the treatment of trees by Donald Knuth, who defines a tree recursively [151, p. 308]: A tree T is a set of ≥ 1 nodes containing one special node designated the *root* of the tree. All nodes except for the root are partitioned into $m \geq 0$ disjointed sets, $T_1, T_2, \ldots, T_m$, each forming a *subtree* of the root. Consider for example the tree shown in Figure 3.3A. By convention the root is drawn at the top, so node A is the root. It has three subtrees, $\{B, E\}$, $\{C\}$, and $\{D, F, G\}$. These are connected to the root by *edges*, sometimes also called *links* or *branches*. The root of subtree $\{B, E\}$ is B, and so on. The *degree* of a node is the number of subtrees rooted on it. *Terminal nodes* or *leaves* have degree 0, while *branch nodes* or *internal nodes* have degree ≥ 1. In Figure 3.3, nodes E, C, F, and G are leaves, the others are internal nodes. It is important to be able to refer to the relative positions of nodes in a tree. A root is called the *parent* of the roots of its subtrees, who are *siblings* and *children* of their parent. In Figure 3.3A, node A is the parent of the siblings B, C, and D. Each node in a tree can be assigned a *level*. The level of each node is one greater than the level of its parent and the level of the root is zero. Our example trees in Figure 3.3 consist of three levels. In an *ordered* tree the order of subtrees matters, while in *oriented* trees it does not. If the two trees shown in Figure 3.3 were ordered, they would be different, while as oriented trees they are equivalent. In this book we shall imply that trees are oriented, unless stated otherwise. An important special kind of tree is the *binary* tree. In it each node has ≤ 2 children. Conventional phylogenies such as the one shown in Figure 1.1 are a special case of binary trees, where all internal nodes have degree 2. As we will see in Chapter 8, there is a further peculiarity about phylogenetic trees: they can be rooted or unrooted (cf. Figure 8.2). This flies in the face of the mathematical definition just given, but corresponds to the usage biologists have established.

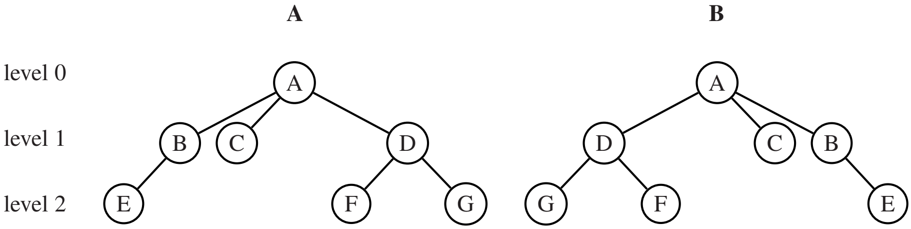


Fig. 3.3. Example trees with different subtree order.

Binary trees are important, because there is a simple method of representing trees with branch nodes of degree > 2 as binary trees. Consider again our example tree Figure 3.3 and remove at each node all references to child nodes except for the first one. Next, insert references connecting groups of siblings, which leaves the total number of links in the tree unchanged. The result of this operation is shown in Figure 3.4A, which is a binary tree. This property becomes more apparent in the equivalent representation Figure 3.4B, with the added rule, that traversal of the link $\swarrow$ changes the level of the node, while traversal of the link $\searrow$ does not.

This brings us to tree traversal. Many algorithms on trees depend on the ability to efficiently visit each node of a tree once. Given a binary tree, tree traversal methods proceed by visiting the root, the left child, and the right child. Depending on the order in which the root is visited, we can distinguish preorder, inorder, and postorder traversal. Preorder traversal proceeds in three steps iterated for each node in the tree:

1. visit node,
2. traverse link to left child,
3. traverse link to right child.

"Visiting" a node in this context means that the algorithm carries out some computation based on the node, while link traversal does no more than that. When applied to the root of the binary version of our example tree (Fig. 3.4B), preorder traversal visits the nodes in the sequence A, B, E, C, D, F, G.

Similarly, inorder traversal proceeds in the following three iterated steps:

1. traverse link to left child,

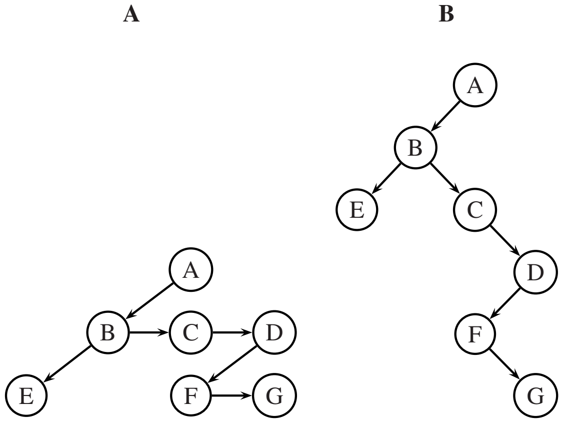


Fig. 3.4. Representation of Figure 3.3A as binary tree. **A**: Generated by removing the child links except for the first one from each node and adding sibling links. **B**: Redrawn version of **A**.

2. visit node,
3. traverse link to right child.

When applied to the root of Figure 3.4B, this procedure visits E, B, C, F, G, D, A. Postorder traversal, finally, proceeds thus:

1. traverse link to left child,
2. traverse link to right child,
3. visit node.

This traversal of Figure 3.4B visits E, G, F, D, C, B, A.

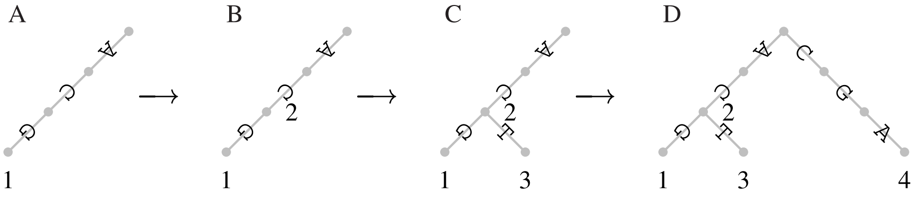
When talking about trees, it is important to be able to visualize them and there are many ways to do this. The explicit style with nodes and edges chosen for Figure 3.3 will be encountered in several variations throughout this book. However, there is a simple representation using nested parentheses, which is often used for textual representations of trees. Transcribing Figure 3.3A in this notation yields $(A(B(E))(C)(D(F)(G)))$.

With these important preliminaries mastered, we are in a position to look at string matching with keyword trees.

3.5 Set Matching Using Keyword Trees

Imagine you wish to establish the location of a set of PCR primers in the human genome. You could sequentially search the genome for every primer sequence. This would take time $O(p \cdot n)$, where p is the total number of primers, i.e. patterns. However, it is possible to make this multiplicative time behavior additive using a keyword tree.

Keyword trees are used for matching a set of patterns against a text and this search method is also known as the Aho-Corasick-algorithm in honor of its inventors [5]. Given a set of patterns, e.g. $\mathcal{P} = \{P_1, P_2, P_3, P_4\}$, where $P_1 = \{\texttt{ACG}\}$, $P_2 = \{\texttt{AC}\}$, $P_3 = \{\texttt{ACT}\}$, and $P_4 = \{\texttt{CGA}\}$, the corresponding keyword tree is constructed by starting with P_1 and a root node connected to a series of child nodes, each one labeled by a single character of P_1 (Fig. 3.5A). The leaf node where the first pattern terminates is labeled 1. The string obtained by concatenating the edge labels on the path from the root to a node is called that node's *path label*. In our example, the path label of node 1 is `ACG`. Continuing with our tree construction we fit $P_2 = \texttt{AC}$ into the existing tree by matching its characters with the characters of the tree, starting at the root. Pattern P_2 terminates at an internal node, and accordingly this node is labeled 2 (Fig. 3.5B). Next, $P_3 = \texttt{ACT}$ is added; it fits until the node 2 and the remainder of P_3 is placed on a new branch. Hence, node 2 has now become a branching node (Fig. 3.5C). Finally, pattern $P_4 = \texttt{CGA}$ is added. Since its first character mismatches the label of the branch emerging from the root, a new branch is added to the root (Fig. 3.5D). This construction algorithm takes time proportional to the combined lengths of the patterns, $|\mathcal{P}| = m$.

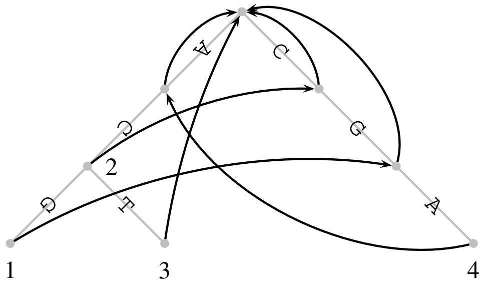


$P_1 = \texttt{ACG}$, $P_2 = \texttt{AC}$, $P_3 = \texttt{ACT}$, $P_4 = \texttt{CGA}$

Fig. 3.5. Construction of keyword tree for pattern set $\mathcal{P} = \{\texttt{ACG}, \texttt{AC}, \texttt{ACT}, \texttt{CGA}\}$.

The keyword tree shown in Figure 3.5 can be used for text searching in the following way: let l be the starting position of a putative pattern in the text and c the currently examined text position. Initially l and c are set to the starting position, usually 1. Start character comparisons at the root and match consecutive characters in the text along the branches of the keyword tree. By doing so, c is increased at every step, while l remains constant. If a labeled node is encountered, report an occurrence of the corresponding pattern starting at position l. If a mismatch is found, increase l by 1, set $c = l$, and start again from the root.

This "naïve" algorithm leads to repeated examination of characters left of c. In order to avoid this duplication of character comparisons, we introduce shortcuts in the keyword tree that enable us to stay below the root even after a mismatch has been encountered. These shortcuts are known as "failure links". Consider node w with path label α. The failure link connects w to that node in the tree whose path label is the longest suffix of α. Figure 3.6 shows our example keyword tree with added failure links. Any failure link from a node whose path label has length 1 must lead to the root. The failure links can be added to the keyword tree during a single breadth first traversal of the tree, leaving the time requirement of keyword tree construction unchanged.



$P_1 = \texttt{ACG}$, $P_2 = \texttt{AC}$, $P_3 = \texttt{ACT}$, $P_4 = \texttt{CGA}$

Fig. 3.6. Keyword tree for pattern set $\mathcal{P} = \{\texttt{ACG}, \texttt{AC}, \texttt{ACT}, \texttt{CGA}\}$ including failure links shown as arrows.

How do the failure links speed up the pattern search? Consider the text `ACGA`. When comparing this to our example keyword tree, we first encounter a hit to pattern 2 and then to pattern 1, before we cannot extend our match any further. In the naïve algorithm we returned to the root at this point and resumed our pattern matching at position 2 in the text. In this case we would examine characters `C` and `G` twice. Instead, we follow the failure link and find a match to pattern 4 with one more character comparison (Fig. 3.6). Careful implementation of this speedup leads to a pattern search in time $O(n + o)$, where o is the number of pattern occurrences [99, p. 60]. If we include keyword tree construction, set matching therefore takes time $O(m + n + o)$.

In many string matching applications the text is not only much longer than the combined pattern lengths, but it is also constant, while the patterns are variable. An example would be searching PCR primer sequences, which are typically between 15 and 20 nucleotides long, in a bacterial genome consisting of the order of 10^6 to 10^7 nucleotides. In this situation it becomes very interesting to have a string matching method that allows searches in time proportional to the length of the pattern, instead of the text.

Turning the speed requirement of string searching on its head by making it independent of the size of the text may sound too good to be true. In fact, we started this chapter by describing a familiar method of achieving this for books: the book's index. Suffix trees are a data structure that achieve the same as well as a number of more advanced string searching tasks.

3.6 Suffix Trees

A suffix tree $\mathcal{T}$ is a special kind of tree associated with a string S, where $|S| = n$. $\mathcal{T}$ has as many leaves as the string has characters and these leaves are labeled $1, \dots, n$. Each edge in the tree is labeled with a substring of S. The defining feature of the suffix tree is that by concatenating the strings from the root along the edges leading to leaf i, the suffix $S[i..n]$ is obtained. Figure 3.7 shows a suffix tree for $S =$ `ACCG`. If you start reading at its root and follow the path leading to leaf 2, you obtain the string `CCG`. This is the suffix that starts at position 2 in S.

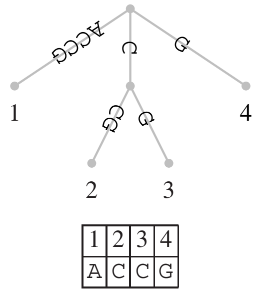
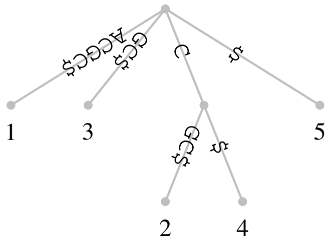


Fig. 3.7. Suffix tree of $S =$ `ACCG`.

In order to search for a pattern P in S, start by comparing the first character in P to the first character of the edge labels emerging from the root of $\mathcal{T}$. If no match is found, we are done and know that P occurs nowhere in S. If, on the other hand, a match is found, follow the corresponding edge label comparing it to successive characters in P. Every time a node is encountered, the correct edge for continuing the string comparison needs to be found. Once a pattern P is detected by this process, we know that S contains an exact match to P. Moreover, the leaf labels in the subtree of the node in whose edge label the match ended indicate the position(s) of P in S. For example, consider pattern $P =$ `C` and $S =$ `ACCG`. If we start matching at the root of the suffix tree shown in Figure 3.7, we end at a node whose subtree contains leaves labeled 2 and 3. These are the start positions of P in S. Looking up the leaves of a subtree takes, to a first approximation, constant time. Given a suffix tree, the exact string matching problem is therefore solved in $O(|P|)$, because at most $|P|$ comparisons between characters in the pattern and characters in the text need to be performed. This time estimate is slightly simplified since at every node encountered during the search the number of necessary character comparisons is less or equal to the size of the alphabet over which S is formed. Nevertheless, pattern matching in an existing suffix tree is extremely fast. This is highly advantageous when dealing with stable databases that are queried frequently, as is often the case in biology.

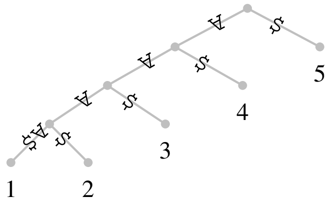
Notice that it is possible that a suffix is a prefix of another suffix. For example, in $S =$ `ACGC` the suffix $S[4..4] =$ `C` is a prefix of the suffix $S[2..4] =$ `CGC`. In this case, no suffix tree corresponding to S exists, as the path labeled $S[4..4]$ would not end at a leaf. A simple method for guaranteeing the existence of a suffix tree is to ensure that the last letter of S does not occur anywhere else in S. In practice, a *sentinel* or *terminator* symbol, often denoted as $\texttt{\$}$ is added to S and $\mathcal{T}$ is constructed from $S\texttt{\$}$ (Fig. 3.8).



1	2	3	4	5
A	C	G	C	$

Fig. 3.8. Suffix tree of $S = \texttt{ACGC}$ including the terminator symbol $\texttt{\$}$.

An extreme case of the identity of prefixes and suffices appears in suffix trees for strings consisting of a single type of character. An example of this is shown in Figure 3.9.

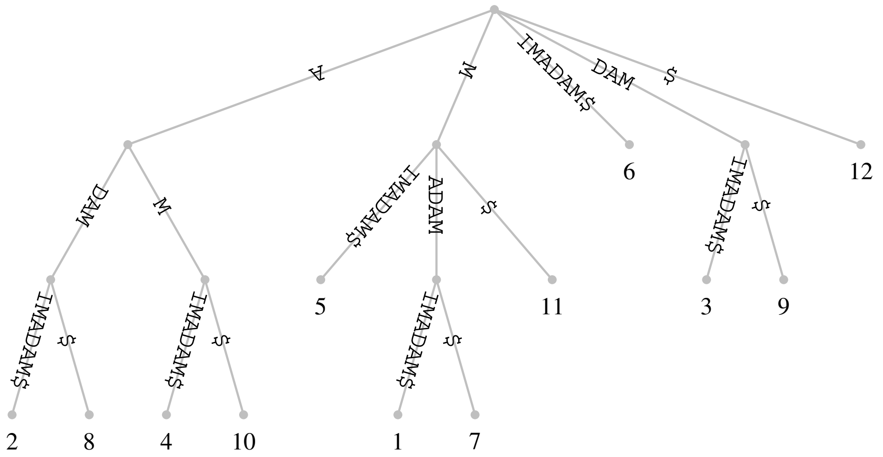


1	2	3	4	5
A	A	A	A	$

Fig. 3.9. Suffix tree of the uniform string $S = \texttt{AAAA\$}$.

Our examples of suffix trees were so far restricted to strings over the nucleotide alphabet. It should be clear that the definition of a suffix tree applies to strings over any alphabet and the suffix tree for the string `MADAMIMADAM` is shown in Figure 3.10.

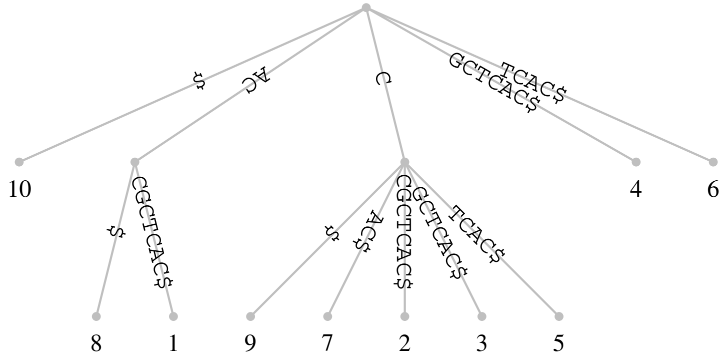
In suffix trees, internal nodes have at least two and at most as many child nodes as the size of the alphabet over which S is constructed. In the case of DNA sequences the corresponding alphabet is $\mathcal{A} = \{\texttt{A}, \texttt{C}, \texttt{G}, \texttt{T}, \texttt{\$}\}$. Hence, the maximal degree of an internal node in a suffix tree is $|\mathcal{A}| = 5$. Figure 3.11 displays the suffix tree


A
M
IMADAM$
DAM
$
DAM
M
IMADAM$
ADAM
$
IMADAM$
$
IMADAM$
$
IMADAM$
$
IMADAM$
$
6
12
5
11
3
9
2
8
4
10
1
7

1	2	3	4	5	6	7	8	9	10	11	12
M	A	D	A	M	I	M	A	D	A	M	$

Fig. 3.10. Suffix tree of $S =$ `MADAMIMADAM$`.

corresponding to the DNA sequence $S =$ `ACCGCTCAC`, which contains an example of an internal node with the maximal degree. Before we explain the uses of suffix trees further, we take a brief look at their construction.


$
AC
C
GCTCAC$
TCAC$
10
4
6
$
CGCTCAC$
$
AC$
CGCTCAC$
GCTCAC$
TCAC$
8
1
9
7
2
3
5

1	2	3	4	5	6	7	8	9	10
A	C	C	G	C	T	C	A	C	$

Fig. 3.11. Suffix tree of $S =$ `ACCGCTCAC$`.

3.7 Suffix Tree Construction

Suffix trees are usually constructed through a series of intermediate trees. One possible naïve algorithm proceeds as follows: Construct a root and a leaf node labeled 1. The edge connecting these two nodes is labeled $S[1..n]$. This initial tree is converted to the final structure by successively fitting the suffices $S[i..n], 1 < i \leq n$, into the intermediate trees. As demonstrated in Figure 3.12, each of these fitting steps proceeds by matching the suffix along the edge labels starting at the root until a mismatch is encountered. Remember that due to the sentinel character a mismatch is guaranteed to occur. Once this is found, there are two possibilities:

1. The match ends at the last letter of an edge label leading to node ν. In this case, an edge labeled with the mismatched part of the suffix is attached to ν. This edge leads to a leaf labeled with the start position of the current suffix. Examples of this are steps **A**→**B** and **C**→**D** in Figure 3.12.
2. The match ends inside an edge label. In this case, an interior node ν is inserted just behind the end of the match. In addition, a leaf labeled i is attached to ν via an edge labeled with the mismatched part of the suffix. An example of this splitting of an edge is step **B**→**C** in Figure 3.12.

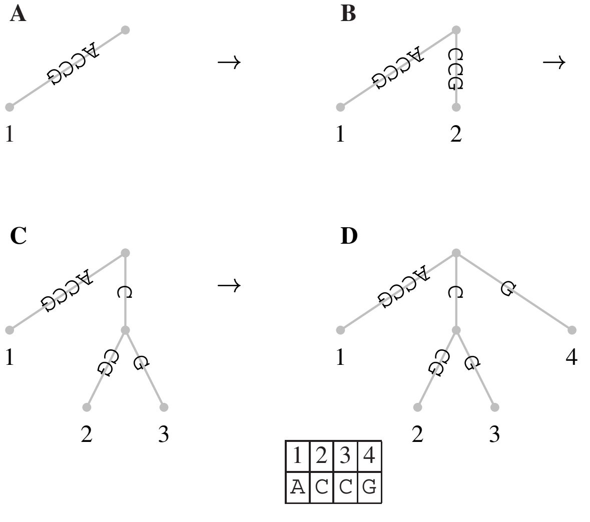


Fig. 3.12. Naïve construction of suffix tree of S = `ACCG`. Starting at the first position and moving leftwards, the suffices of S are successively fitted into the tree.

This naïve algorithm runs in $O(|S|^2)$ time, which makes it impractical for realistic applications. Fortunately, there are three principal algorithms available that run in linear time [177, 249, 259]. In other words, it is feasible to construct suffix trees with the same time behavior as the Z-algorithm introduced in Section 3.3. However, we have already seen that once such a suffix tree is available, it can be searched repeatedly in time proportional to the length of the pattern.

In spite of the availability of linear-time construction algorithms, suffix trees come at some cost in both time and space. Since computer memory is often more limiting than computer time, we now take a brief look at the space issue.

3.8 Suffix Arrays

A suffix tree has a maximum of $2n-1$ nodes; its space requirement is therefore linear in the length of the text and we might be tempted to leave it at that. However, suffix tree nodes take up at least ten times more computer memory than a simple character [158]. Moreover, for efficient string searching the complete suffix tree usually needs to be held in the random access memory (RAM) of a computer.

A more space-efficient representation of some of the information contained in a suffix tree is captured in a suffix array [172]. This consists of the lexical ordering of a string's suffices. Such an ordering can easily be obtained from a suffix tree if edges are lexically ordered. For example, the edges of the suffix tree in Figure 3.11 are lexically ordered from left to right. A depth-first traversal of this tree retrieves the suffices depicted in Table 3.1 in lexical order. Since a bare-bones suffix array consists solely of a one-dimensional array holding the second column of Table 3.1, it is very space efficient. In practice, suffix arrays are reported to take up three to five times less space than suffix trees [172].

Table 3.1. Lexical ordering of the suffices contained in $S =$ `ACCGCTCAC`. This can be obtained by noting the leaf labels during an inorder traversal of the suffix tree depicted in Figure 3.11. A suffix array simply consists of the suffix start positions uncovered in this fashion.

index	suffix start position	suffix
1	8	`AC$`
2	1	`ACCGCTCAC$`
3	9	`C$`
4	7	`CAC$`
5	2	`CCGCTCAC$`
6	3	`CGCTCAC$`
7	5	`CTCAC$`
8	4	`GCTCAC$`
9	6	`TCAC$`

How can a suffix array be used for string matching? Since the array is ordered, it can be searched using a technique known as binary search. Such a search starts at the midpoint of the array and determines whether the corresponding suffix's lexical position is greater or less than that of the string concerned. In this way half of the array can be discarded for further search and this halving of the search space is repeated successively until a match is found. In the case of suffix arrays, the position of the highest and of the lowest match in the array should be searched for. These boundaries then delineate all positions at which the desired pattern occurs. If for example, we wish to know where in $S = \texttt{ACCGCTCAC}$ pattern $P = \texttt{C}$ occurs, we find the lowest matching position at index 3 and the highest matching position at index 7. The corresponding positions in the string are 9, 7, 2, 3, and 5.

The utility of suffix trees extends beyond rapid string matching. One area of particular interest is their application to the detection of repetitive sequences, which are an important feature of the genomes of complex organisms.

3.9 Repetitive Sequences in Genomics—the C-value Paradox

The amount of DNA found in the haploid genome of an organism is called its C-value. Given that a genome encodes the organism of its host, it is perhaps surprising that there is little correlation between the C-value and an organism's complexity: The size of the genome of the unicellular baker's yeast *Saccharomyces cerevisiae* is $1.2 \cdot 10^7$ base pairs (bp) [88], which is roughly 200 times smaller than that of humans ($3.1 \cdot 10^9$ bp) [128], whose genome in turn is approximately 200 times smaller that that of the unicellular amoeba *Amoeba dubia* ($6.7 \cdot 10^{11}$ bp) [168, p. 383]. This lack of correlation between complexity and genome size in eucaryotes is an observation first made in the late 1960s [27] and is known to biologists as the C-value paradox [201]. The C-value paradox is mainly caused by the varying amounts of repetitive DNA found in metazoan genomes. Repetitive sequences can be classified into five categories [128, p. 879ff]:

1. transposon-derived repeats; these are also referred to as interspersed repeats and make up the vast bulk of repetitive DNA (Table 3.2); they vary in length between 100 and 6,000 bp and in the majority of cases are replicated through an RNA intermediate;
2. inactive retroposed copies of cellular genes, also referred to as *processed pseudogenes*;
3. simple sequence repeats, e.g. $(\texttt{A})_\text{n}$ or $(\texttt{GC})_\text{n}$, also known as microsatellites;
4. segmental duplications, which are 10 kb-300 kb chunks of DNA copied from one region of the genome to another;
5. blocks of tandemly repeated sequences, e.g. centromeres, telomeres, and ribosomal gene clusters.

The function of repetitive DNA remains unclear [201]. However, these elements make assembly of shotgun fragments (cf. Section 2.7) difficult, which is illustrated in Figure 3.13. This is a particularly tricky issue in the automated assembly of eucaryotic genomes [180]. On the other hand, repetitive sequences have come to play an important role in a number of molecular techniques. For example, microsatellites are highly variable and for this reason are used as markers in mapping and population

Table 3.2. Interspersed repeats as percentage of the genome of various multicellular eucaryotes [128, 40].

organism	trivial name	interspersed repeats
Homo sapiens	human	44.4%
Mus musculus	mouse	38.6%
Arabidopsis thaliana	thale cress	10.5%
Caenorhabditis elegans	nematode worm	6.5%
Drosophila melanogaster	fruit fly	3.1%

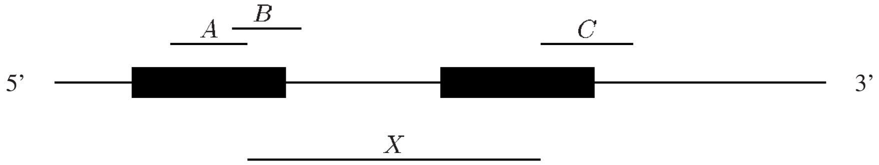


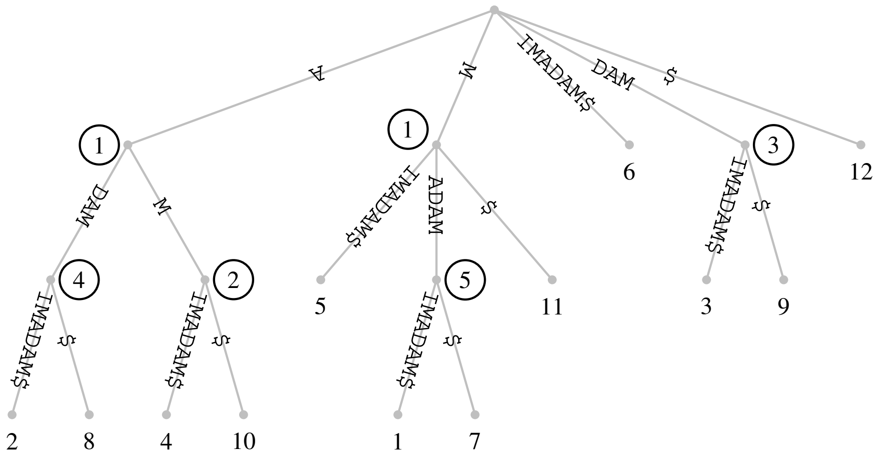
Fig. 3.13. Automatic genome assembly is made difficult by repetitive sequence elements shown as boxes along a chromosome (line). In our example the 3' end of fragment A can either overlap with fragment B or with fragment C. Joining fragments A and B based on their overlap results in the correct assembly. In contrast, joining fragments A and C leads to the deletion of the genomic region X.

studies. Further, the precise distribution of the most prevalent interspersed repeat in humans, the *Alu* element, forms the basis of DNA fingerprinting used in forensics. Thus, dealing with repetitive sequences is very much a feature of modern molecular biology. These repeats are usually inexact. They are routinely detected through aligning known repeat elements to the genome of interest. However, such inexact repeat elements often lead to an increased prevalence of exact repeats. In the following section we give a first introduction to suffix-tree-based algorithms for detecting exactly repeated as well as unique substrings.

3.10 Detection of Repeated and Unique Substrings Using Suffix Trees

We have already seen in Section 3.6 that suffix trees can be used for rapid string matching. A corollary of the way in which this matching is carried out is that any prefix of an internal node's path label is a repeat substring. The suffix tree depicted in Figure 3.11 has two internal nodes with path labels `AC` and `C`. These substrings together with the prefix `A` of `AC` are the only three repeats contained in the underlying text `ACCGCTCAC`: `AC` and `A` are both repeated twice (positions 8 and 1), while `C` is repeated five times (positions 9, 7, 2, 3, and 5).

We define the *string depth* of an internal node as the length of its path label. For example the node with string depth 5 in Figure 3.14 has path label `MADAM`, which consists of five characters. String depths can be assigned to internal nodes in a single depth first traversal. By looking up the internal node with, say, the greatest string depth, we can rapidly locate the longest exact repeat in a string. Moreover, we have already learned that, apart from the single root, trees only contain two kinds of nodes: internal nodes and leaves. Hence a path label either ends at an internal node, in which case it is a repeat substring, or at a leaf, in which case it is a unique substring.



1	2	3	4	5	6	7	8	9	10	11	12
M	A	D	A	M	I	M	A	D	A	M	$

Fig. 3.14. Suffix tree of $S =$ `MADAMIMADAM$` with string depths marked as circled numbers.

Unique sequence motifs are perhaps as important as repeat motifs in molecular biology. They play a role in the construction of PCR primers, the molecular identification of organisms, and the generation of specific antibodies. In natural sequences the number of unique sequence motifs tends to be of the same order of magnitude as the total number of substrings, i.e. $O((n + 1)n/2) \approx O(n^2)$ for a sequence of length n. As n grows, the number of candidate unique sequences quickly becomes unmanageable. Here suffix trees suggest a way of reducing a problem of quadratic complexity to its linear version: Visit each leaf in a suffix tree and look up its parent's string depth. This number plus one is the minimal length of a unique string starting at the leaf's position. There are $O(n)$ such shortest unique substrings from which all the remaining unique substrings can be generated by simple extension. Consider for example the suffix tree shown in Figure 3.14 and say that the first leaf we visit is 2. The string depth of its parent node is 4 and hence we know that the shortest unique substring starting at position 2 has length $4 + 1 = 5$. This is `ADAMI` and any extension of this string to the righthand side such as `ADAMIM`, `ADAMIMA`, etc. is unique.

3.11 Maximal Repeats

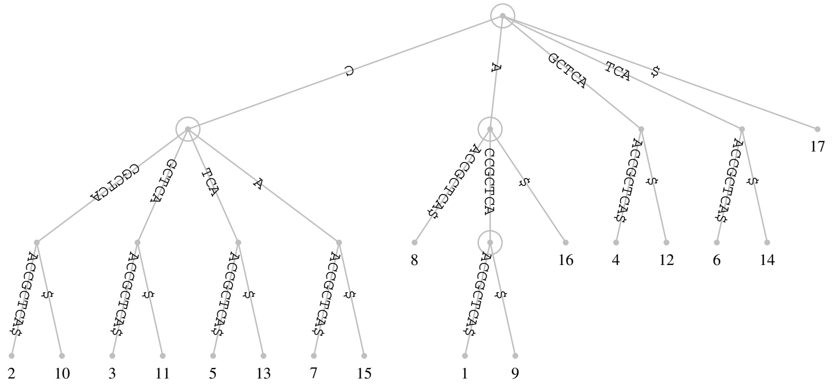
As with unique substrings, there are usually very many repeat motifs contained in a given sequence, for instance, about 1/4 of all positions in the genome of *Escherichia coli* are occupied by `A`. In practice we are therefore often most interested in finding *maximal repeats*. These have the property that they become unique when extended either to the left or to the right. For example, the string `TACACT` contains a maximal repeat of length 1 (`T`) and a maximal repeat of length 2 (`AC`). In contrast, the substring `A` is a repeat, but not a maximal repeat.

The central idea for the detection of maximal repeats is to look up the left neighbors of the leaf positions in the subtree of some internal node ν. For example, the suffix tree for $S =$ `ACCGCTCAACCGCTCA` is shown in Figure 3.15. Its rightmost internal node has string positions 6 and 14 in its subtree. The left neighbor of both positions is $S[5] = S[13] =$ `C`. If in contrast to this example the left neighbors in the subtree rooted on ν are occupied by more than one character, ν is designated *left-divergent* [99]. The property of left-divergence propagates up the suffix tree, i.e. a node with a child node that is left-divergent is itself left-divergent. The path label of any left-divergent node corresponds to a maximal repeat. The path label of the left-divergent node with the greatest string depth is the longest maximal repeat in S. In our example suffix tree (Fig. 3.15) all left-divergent nodes are circled and you should verify that the longest maximal repeat in S is `ACCGCTCA`, which is the path label of the left-divergent node with the greatest string depth. It occurs in S at positions 1 and 9.

3.12 Generalized Suffix Tree

So far, we have constructed suffix trees just from a single string. However, it is straight forward to generate what is known as a *generalized suffix tree* representing more than one string. Consider the two strings S_1 and S_2. We can proceed by first constructing the suffix tree for S_1 and then inserting into this every suffix of S_2. All we need to do is change the leaf labels such that they now consist of pairs of numbers identifying both the string of origin and the position within that string. As an example, consider $S_1 =$ `ACCGCTCAC`; the corresponding suffix tree is shown in Figure 3.11. If we insert into this suffix tree $S_2 =$ `GCA`, we obtain the tree topology shown in Figure 3.16.

Using the new labeling scheme for leaves, generalized suffix trees of an arbitrary number of strings can be constructed. This opens the way for solving the longest common substring problem.



1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
A	C	C	G	C	T	C	A	A	C	C	G	C	T	C	A	$

Fig. 3.15. Suffix tree for $S =$ `ACCGCTCAACCGCTCA$`. The circled nodes are left-divergent (see text for details). The left-divergent node with the greatest string depth is labeled by the longest maximal repeat in S: `ACCGCTCA`.

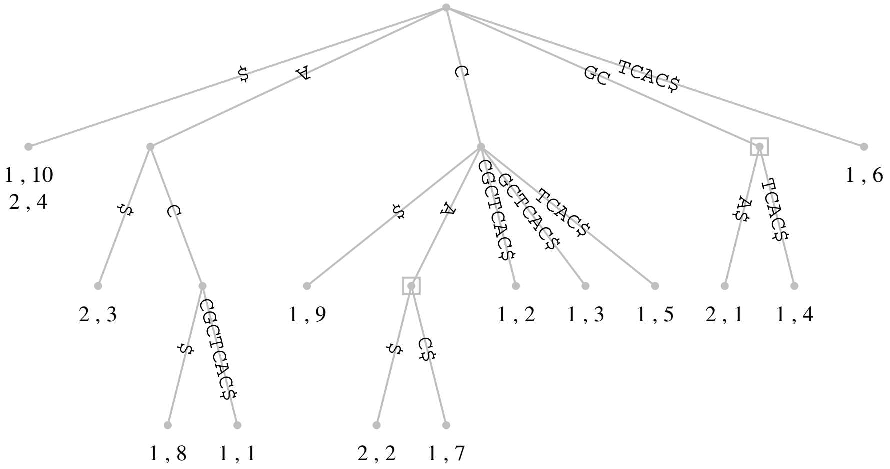
3.13 Longest Common Substring Problem

The longest common substring problem is motivated by elementary biological considerations. When comparing two or more coding DNA sequences, regions that are well conserved are often of particular biological interest. They might encode functionally important domains of the corresponding protein or point to novel regulatory elements. In addition, they represent suitable sites for PCR-primers that work across all of the sequences considered. So there are many occasions when we wish to know the longest substring shared by all members of a set of strings. For example, the two longest common substrings between $S_1 =$ `ACCGCTCAC` and $S_2 =$ `GCA` are `GC` and `CA`.

To find the longest common substring between our example sequences $S_1 =$ `ACCGCTCAC` and $S_2 =$ `GCA`, we need to find the *lowest common ancestor* of leaves containing a suffix from S_1 and a suffix from S_2. One way to do this is to note for each internal node the string identifier of the leaves in its subtree. Those nodes that are marked 1 and 2 end at a path that is labeled by a substring common to S_1 and S_2. All we need to do now is search for the internal node marked by 1 and 2 that has the greatest string depth. In Figure 3.16 these are the two boxed nodes.

3.14 k-Mismatches

The examples presented so far have given a first impression of the usefulness of suffix trees in the context of exact string matching. However, in biology we are often more



	1	2	3	4	5	6	7	8	9	10
S_1	`A`	`C`	`C`	`G`	`C`	`T`	`C`	`A`	`C`	`$`
S_2	`G`	`C`	`A`	`$`						

Fig. 3.16. Generalized suffix tree for the strings S_1 = `ACCGCTCAC$` (cf. Figure 3.11) and S_2 = `GCA$`. The leaf labels consist of pairs of numbers (i, j), where i denotes the string of origin, and j the suffix position within that string. Notice that leaves may carry as many labels as there are input strings. The path labels of the two boxed nodes are the longest common substrings between S_1 and S_2.

interested in *similar* rather than identical sequences. It turns out that the concept of the lowest common ancestor first introduced in the context of searching for longest common substrings (Section 3.13) is also useful for extending the application of suffix trees to *inexact matching*. A standard version of the inexact matching problem is known as the k-mismatch problem. Given a string S, the task is to find all positions of pattern P with at most k mismatches. Now, the string depth of any lowest common ancestor for two leaves pointing to positions i and j in S indicates the length of the longest match starting in positions i and j. The same observation applies if we build the generalized suffix tree for S and P; in this case the string depth of a lowest common ancestor for position i in S and position j in P indicates the length of the match starting at $S[i]$ and $P[j]$. So we start by looking for the lowest common ancestor between leaves labeled $S[1]$ and $P[1]$. If its string depth, l, is equal to $|P|$, we have found an exact match of P at position $S[i]$. On the other hand, if $l < |P|$, repeat the procedure by looking for the lowest common ancestor for leaves pointing to $S[i + l + 1]$ and $P[l + 2]$. If at most k repetitions of this step are sufficient to cover P, a match has been found starting at position i. Lowest ancestor queries can be carried out in constant time [99, ch. 8]. Therefore, generalized suffix trees allow the detection of k-mismatches in time $O(k \cdot |S|)$. In other words, by allowing only a single mismatch, the runtime of our string search changes from being proportional to the length of the pattern to being proportional to the generally much greater length of the text.

In order to use this procedure to find repeats between $S_1 = \texttt{ACCGCTCAC}$ and $S_2 = \texttt{GCA}$ with at most $k = 1$ mismatch, we can return to the generalized suffix tree depicted in Figure 3.16. We start at the leaves labeled $1,1$ and $2,1$; the corresponding lowest common ancestor is the root, whose string depth is zero. So the next step is to look at the leaves labeled $1,2$ and $2,2$. The string depth of the lowest common ancestor is 1, which means that $S_2[3]$ is mismatched. Since $S_2[1]$ and $S_2[3]$ are mismatched, we conclude that there is no 1-mismatch of S_2 at $S_1[1]$. This procedure is repeated until the lowest common ancestor of the leaves labeled $1,4$ and $2,1$ is visited (boxed node in Figure 3.16). Its string depth is 2 and, hence, we have found a 1-mismatch occurrence of S_2 at $S_1[4]$. Another 1-mismatch occurrence of S_2 starts at $S_1[7]$.

Notice that the inexact matching we have carried out so far does not encompass insertions and deletions (indels). However, these are an integral feature of sequence evolution and need to be accounted for in many biological contexts. Indels cannot be handled easily using suffix trees. As explained in Chapter 2, string comparisons incorporating indels are best done using alignments based on dynamic programming. We shall see in Chapter 4 that a combination of dynamic programming and exact matching techniques leads to very fast and memory-efficient alignment algorithms.

3.15 Summary

In computational biology one often needs to look up the occurrence of some pattern P in a text T. Since the texts of computational biology include genome sequences, which tend to be large, it is important to apply efficient methods of string matching. Traditional string matching methods are guaranteed to take time $O(n)$, where n is the length of the text. By preprocessing a set of patterns into a keyword tree, this time requirement can be extended to set matching. Instead of preprocessing one or more patterns, it is also possible to preprocess the text. A suffix tree is a data structure that can be constructed for a given text in $O(n)$. However, once it is constructed, it can be used to search any P in T in time $O(m)$, where m is the length of the pattern. In addition to making string searching extremely efficient, a suffix tree reveals in one fell-swoop the entire repeat structure of T without the need for carrying out any string comparisons. This has important biological applications where unique and repeat sequences play a central role in many fundamental as well as biotechnological problems. Finally, suffix trees can also be used for rapid inexact string matching, where $\leq k$ mismatches between P and its occurrence in T are allowed.

3.16 Further Reading

Gusfield has written a comprehensive and detailed guide to string algorithms in computational biology [99].

3.17 Exercises and Software Demonstrations

3.1. Suffix tree construction is relatively time consuming while searching a suffix tree is quick. Use the program `Match`→`String Matching` in the `bioinformer` software (Section A.2.1) to see how this assertion compares with timings of real-world string searches.

3.2. Generalized suffix trees provide a data structure for efficiently finding longest exact matches between two sequences. Construct an algorithm based on dynamic programming for achieving the same end. Compare the run times of the two approaches.

3.3. Draw the suffix tree corresponding to $T = \texttt{AAGCG}$ and compare your result with that calculated using the program `Match`→`Suffix Tree` of the `bioinformer` software (Section A.2.2).

3.4. Suffix trees are ideal for determining exact repeats in strings. The program `Match`→`Repeat Search` in the `bioinformer` software (Section A.2.3) implements a search for repeat sequences based on a suffix tree. Use this program to find the longest repeat sequence in the first genome to be completely sequenced—that of bacteriophage ϕX174 [214]—as well as the longest repeat sequence in the paper in which James Watson and Francis Crick first proposed the doublehelical structure of DNA [256].

4

Fast Alignment: Genome Comparison and Database Searching

> *In the process of routinely screening new sequences having no known relatedness to previously determined protein sequences, we noted regions of similarity of the catalytic chain of bovine cAMP-dependent protein kinase (BOV-PK) to the inferred amino acid sequences of the src gene products (transforming proteins) of Moloney murine sarcoma virus (MMSV) and of Rous avian sarcoma virus, Schmidt-Ruppin strain (RSV-SR).*
>
> *Winona C. Barker and Margaret O. Dayhoff [16, p. 2836]*

Biology has traditionally been a comparative science and computational molecular biology has inherited this approach in the form of a strong emphasis on sequence comparison. If genomic regions of closely related organisms are available, a full alignment of these is conceptually a simple extension of the global alignments of genes that were a staple of the gene-centered pre-genomic era of molecular biology. However, pairwise alignment by dynamic programming as introduced in Chapter 2 has a space and time requirement of $O(m \cdot n)$, where m and n denote the lengths of the sequences compared. This does not scale well.

In order to tackle the memory requirement of optimal alignment algorithms, observe that in order to fill the corresponding dynamic programming matrix, the algorithms first encountered in Section 2.6 need only save one row (or column) plus one additional cell. Starting from such a configuration (Fig. 4.1A), the algorithm stores the value of a new cell and erases a cell that is not needed in subsequent steps (Fig. 4.1B). This is repeated until the last cell in the row has been filled (Fig. 4.1C) and the circle can start again (Fig. 4.1D). Since the dynamic programming matrix can be filled either column-wise or row-wise, the maximum score of an alignment can thus be calculated in space proportional to the shorter of the two sequences being aligned. However, in most cases we are interested in obtaining the actual alignment rather than simply its score. Based on the idea of computing the score in linear space, an elegant algorithm has been devised that returns the alignment in linear space, while leaving the time requirement proportional to the size of the original dynamic programming matrix [116, 181]. This is a remarkable advance, as the memory requirement of an algorithm may pose an insurmountable barrier, whereas a scientist should never run out of patience. Still, with exponentially growing databases speed is important.

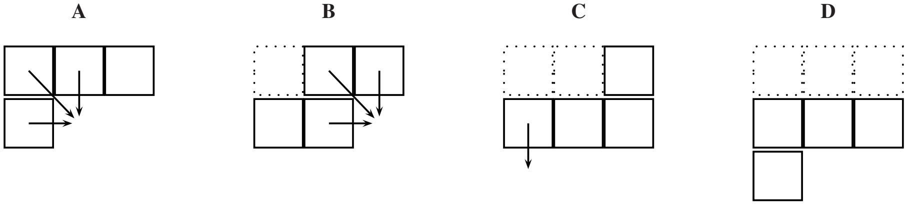


Fig. 4.1. A–D: Filling in the dynamic programming matrix using the space corresponding to just one row plus one cell. Solid cells are active, dotted cells have been deleted.

For improving the run time of global alignment algorithms (as opposed to their memory requirement), a more radical departure from dynamic programming is necessary. One possible approach developed for aligning genomes of closely related bacteria is discussed in this chapter. It involves a combination of suffix trees introduced in Chapter 3 and dynamic programming introduced in Chapter 2.

On the other hand, initial annotation through homology involves finding a local inexact match between a short *query* sequence and a large number of *subject* sequences that make up the *text* of the search. In this case we are looking for one or more local alignments between the query sequence and the database entries. These databases may be very large. For instance, the GenBank sequence repository has been growing exponentially since it was started in 1982 (Fig. 4.2). Searching GenBank by dynamic programming is as impractical as in the case of globally aligning bacterial genomes. However, the task is also of a different kind, as we expect that the vast majority of entries in the database are of no interest in our search. Hence most contemporary algorithms for searching biological sequence data bases aim to filter or exclude these irrelevant regions before committing to a more detailed (and time-demanding) investigation of regions that might contain a match.

In addition to algorithmic problems, database searching also poses statistical challenges. When finding an alignment between a query sequence and an entry in a potentially very large database, it is important to know how likely this match would occur by chance alone.

In this chapter we first give an example of doing biology by fast global genome comparison before explaining the algorithmic issues involved. This is followed by two examples of using rapid local alignments as a guide to biological function, which leads into an explanation of the corresponding computational ideas. Finally, an application of fast local alignment to the problem of discovering the extent of genome-wide gene duplication motivates our treatment of the statistical issues involved in database searching.

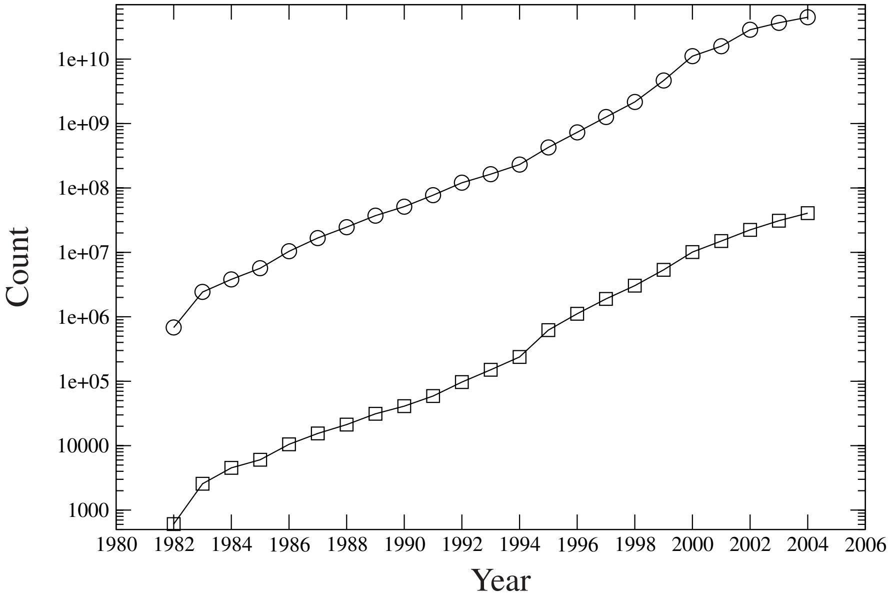


Fig. 4.2. The GenBank sequence repository has grown exponentially since its inception in 1982. ○: residues (nucleotides and amino acids); □: entries. Data taken from `www.genome.jp/dbget/db_growth.html`.

4.1 Global Alignment

Fast global alignment is often used for comparing the genomes of closely related organisms. This is particularly interesting in organisms where the relationship between genotype and phenotype is comparatively easy to investigate such as in bacteria. Many pathogenic bacteria have been subjected to comparative genome sequencing. For instance, *Vibrio parahaemolyticus* is a gram-negative marine bacterium and a frequent cause of food poisoning among people fond of seafood. A pandemic of gastroenteritis caused by this organism prompted the sequencing of its genome in 2003 [171]. In order to learn more about its virulence mechanism, the resultant sequence of its two chromosomes, one 1.8 Mb and the other 3.3 Mb in length, was compared to the genome sequence of the related bacterium *Vibrio cholerae*. The latter causes cholera, a more severe epidemic diarrhea. Like *V. parahamolyticus*, the genome of *V. cholerae* is organized in two chromosomes. One of the tools for carrying out the genome comparison was the software `MUMmer` [48], which rapidly computed a global alignment of the larger of the two chromosomes [171]. The researchers were surprised to find distinct virulence mechanisms in this comparison, even though both bacteria cause similar diseases. *V. parahaemolyticus* contains a complex of genes known as the type III secretion pathway, which is common among diarrhea-causing pathogens including shigella, salmonella, and enteropathogenic *Escherichia coli*. *V. cholerae*, on the other hand, does not contain the type III secretion pathway. Such comparative information on bacterial pathogenicity obtained by genome alignment is likely to impact future treatment of the corresponding diseases [171].

How does `MUMmer` achieve efficient global alignment on a genomic scale?

Strategy for Fast Global Alignment

In order to make direct pairwise comparisons between complete genomes feasible, `MUMmer` is based on a generalized suffix tree [48]. Recall from Section 3.12 that a suffix tree for a single sequence can be generalized to index any number of sequences. Here, we are interested in comparing only two sequences (genomes). In the corresponding *generalized* suffix tree exact matches between the two sequences are marked by internal nodes whose subtrees contain entries from both sequences. `MUMmer` carries out three steps to align whole genomes [48]:

1. identify all ***m**aximal **u**nique **m**atches* (MUMs) between the two genomes; such repeats cannot be extended and occur only as a single pair of matching substrings between and not within the genomes;
2. find the longest increasing subsequence of MUMs;
3. process the gaps in the resulting alignment.

We now explain these steps further. Notice first of all that—apart from the uniqueness criterion—MUMs correspond to the maximal repeats introduced in Section 3.10. These can be found in a suffix tree in linear time. The intuition behind searching for these maximal repeats is that they are highly likely to form part of the final global alignment.

To fix our ideas, let us represent a MUM by the triplet (i, j, l), where i is the start position in genome A, j the start position in genome B, and l the length of the repeat. We can sort the triplets representing these fragments according to their position in, say, genome A. This induces a labeling on the matched fragments in genome B, which is not necessarily monotonously increasing (Fig. 4.3, top panel).

The next step in the algorithm is to find the longest increasing subsequence of MUMs from genome B. This may well lead to the loss of fragment pairs. For example, in the bottom panel of Figure 4.3 fragment 4 has been removed from the set of MUMs in both genomes.

The longest increasing subsequence of MUMs usually contains gaps which need to be filled next. Depending on the type of gap, their processing varies: large insertions are simply marked graphically in the final alignment, while polymorphic regions are subjected to alignment by dynamic programming. Since the two genomes that are being compared are assumed to be closely related, the time spent on gap closure by dynamic programming should be negligible [48].

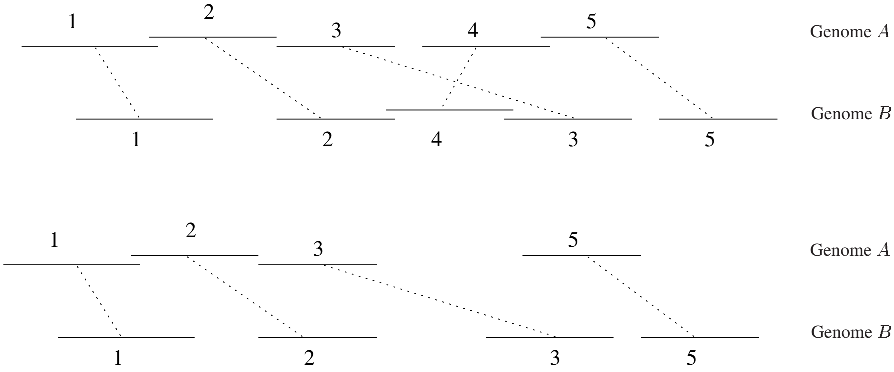


Fig. 4.3. Sorting step in the algorithm for constructing global pairwise alignments of whole genomes. *Top panel*: Maximal unique matches (MUMs) sorted according to position in genome A. MUM 4 has been transposed. *Bottom panel*: Longest increasing subsequence extracted from the MUMs of genome B. Notice that in our example the longest increasing subsequence is not unique, $1, 2, 4, 5$ would also be a valid subsequence. The configuration of MUMs induced by the longest increasing subsequence is finally subjected to gap processing. (Redrawn from [48].)

Recall that alignment by standard dynamic programming is based on a rather simple evolutionary model which encompasses only mutations and short indels. Genomes, however, may be subject to large-scale changes due to, for example, transposition or horizontal gene transfer. The approach to global alignment presented above can account for such changes. In addition, it runs in time linear in the combined length of the input sequences if these are closely related [48].

Fast global alignment algorithms are set to play an increasing role as genome data from closely related organisms accumulates [110]. However, annotation of individual genes contained in these genomes is usually carried out using methods for rapidly detecting local alignments.

4.2 Local Alignment

Rapid local alignment methods are often applied in the context of searching biological sequence databases. One of the earliest examples of how such a database search can illuminate a protein's function is connected to a seminal discovery in medicine: the pathogenicity of cancer-causing viruses is due to their expression of genes derived from homologous genes expressed in healthy cells. This astonishing relationship between virology, cancer, and normal cell proliferation has its origins in work started early in the last century. In 1911, the American Virologist Peyton Rous described in chicken the first infectious agent capable of causing cancer. Fifty-five years later Rous was awarded the Nobel Prize for his discovery of the "Rous Sarcoma Virus" (RSV) as it had become known by then. The genome of RSV consists of a single-stranded RNA molecule comprising 9,392 nucleotides and by the time of Rous' Nobel Prize in 1966 it was speculated that this RNA molecule was reverse transcribed into a DNA molecule, which then integrates into the genome of its host [251, 23]. In 1970, David Baltimore and Howard Temin published independently the discovery of an enzyme that could affect this reverse flow of genetic information: the reverse transcriptase. RSV thus became the archetypical *retrovirus*, a group of morphologically similar viruses that also contains the human immunodeficiency virus (HIV). In the year after the discovery of reverse transcription, Peter Vogt isolated a mutant that had lost the ability to transform its host cells into neoplastic cells, but that replicated normally. This transformation-deficient yet replication-competent mutant of RSV could be compared by hybridization to its wildtype, leading to the isolation of the sequence-fragment that caused the difference between the two. This fragment was then used as a probe specific for the gene, which became known as *src*. The protein product of this gene was isolated in 1977 and its activity as a tyrosine protein kinase was established by 1980. Protein kinases play a central role in intracellular signaling, which explains how a single transforming gene can cause the many changes involved in the transition from normal cell cycle to neoplastic growth. Intriguingly, a protein indistinguishable from the *src* protein was soon discovered in healthy, uninfected cells. It became evident that the virus had captured a cellular gene sometime in the evolutionary past. This gene had remained so well conserved that it could be detected by successfully annealing a *src*-specific DNA probe from chicken to the genomes of other vertebrates and even members of other phyla, including *Drosophila* and *Hydra* [251, 23].

Still, the cellular identity of the *src* gene remained elusive. In 1981 the protein sequence of a bovine tyrosine kinase was published. This was compared to the 1,560 other protein sequences known at the time, revealing that it was similar to *src*. The immediate conclusion from this computerized database search was that the sequenced mammalian tyrosine kinase was a proto-oncogene, i.e. a gene that occurs in normal cells but may cause cancer upon changes in its expression level, or due to certain point mutations. Unfortunately, the normal function of this gene remained unknown at the time, although the authors speculated that it might play a central role in normal growth and development [16].

The feat of assigning a function to a proto-oncogene through mere database searching was achieved two years later by Russell Doolittle and his colleagues [50]. They showed that the transforming protein of another tumor retrovirus, the simian sarcoma virus, was homologous to the sequence of human platelet-derived growth factor (PDGF). This protein is stored in platelets and released into the serum in response to wounding. PDGF contributes to wound healing by stimulating cell division and it is not difficult to imagine how constitutive over-stimulation of cell division might lead to neoplastic growth.

Today, database searches for annotating sequences form a routine part of research in molecular biology. It turns out that the various programs available for this task are based on the common algorithmic idea of basing slow, inexact string matching on rapid, exact matching.

4.2.1 Global/Local Alignment: k-Error Matching

In Section 3.14 we first mentioned the k-mismatch problem, which is solved by finding all occurrences of pattern P (length m) in text T (length n), such that an occurrence has at most k mismatches. We now return to this problem and expand its biological applicability by not only allowing mismatches but also indels.

Let k denote the maximum number of *errors* (mismatches or indels) allowed in a match. The crucial observation for the design of fast alignment programs is that an occurrence of P in T implies that there exists at least one substring of P with a perfect match of length $\geq r$ in T. The length of r can easily be computed by dividing P into $k+1$ contiguous substrings of equal length:

$$r = \frac{m}{k+1}. \tag{4.1}$$

Figure 4.4 illustrates this idea for $k = 5$. If we divide the query into $5+1 = 6$ segments of equal length and randomly distribute the errors along the query, at least one of the fragments remains error free [15].

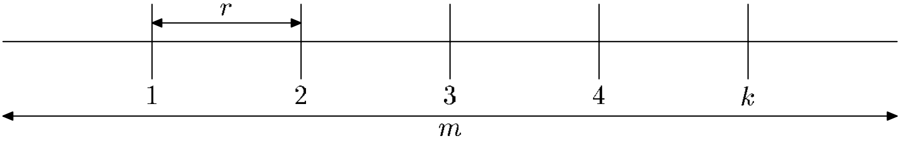


Fig. 4.4. An occurrence of pattern P, $|P| = m$, with at most k errors, implies that P contains a perfectly matching substring of length $\geq r = m/(k+1)$.

The important insight here is that an exact match of non-trivial length is a necessary condition for an inexact match. The resulting algorithm is divided into two phases, a search and a checking phase. At the beginning of the search phase the pattern is divided into $k+1$ contiguous segments of equal length. T is then searched for perfect matches to these segments. One rapid way of doing this would be to use a suffix tree of T and search this $k+1$ times. As a more memory-efficient alternative, the $k+1$ substrings of P might be preprocessed into a keyword tree (cf. Section 3.5).

For a random pattern and text with equiprobable characters, the expected number of matches found in the search phase, E_{match}, is equal to the probability of finding a match of length r times the length of the text, times the number of starting points for fragments of length r:

$$E_{\text{match}} = \frac{1}{|\mathcal{A}|^r} \cdot (k+1)n = \frac{(k+1)n}{|\mathcal{A}|^r}$$

where $|\mathcal{A}|$ is the size of the alphabet.

In the checking phase the matches returned by the search phase are processed further. If a match is found starting at positions $T[i]$ and $P[j]$, the algorithm examines substring $T[i-j+1-k..i+m-j+k]$ for a match using dynamic programming. Figure 4.5 illustrates this reasoning. Here $k = 4$, while the length of m is only given in units of r. The algorithm is designed for the case where $k \ll r$ and hence we can think of r as, say, 10^2 bp. The choice of the substring of T to be aligned

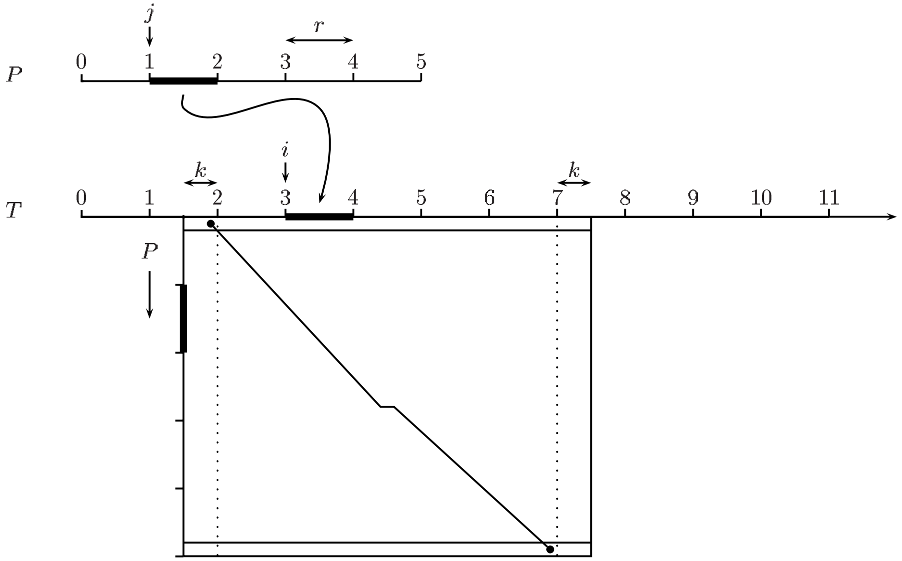


Fig. 4.5. The principle of k-error matching. The bold segment of length r starting at position j in the pattern P forms an exact match with the bold segment at position i in the text T. A substring of the text spanning the length of the pattern plus the maximal error at each margin is aligned with the pattern. The traceback starts at the cell with the maximum entry out of the rightmost $2k$ cells and stops upon reaching the top row. In case $> k$ errors are encountered along the path, the traceback is aborted. Not drawn to scale.

with P means that there are $m(m+2k)$ dynamic programming operations for each exact match found during the search phase. Therefore, the algorithm runs in time proportional to

$$m(m+2k)E_{\mathrm{match}}.$$

In other words, the algorithm runs in time $O(m^2 \cdot n)$. This might look like a disappointing result given that straight dynamic programming takes time $O(m \cdot n)$. However, as long as the number of exact matches of length r is small, the algorithm is fast.

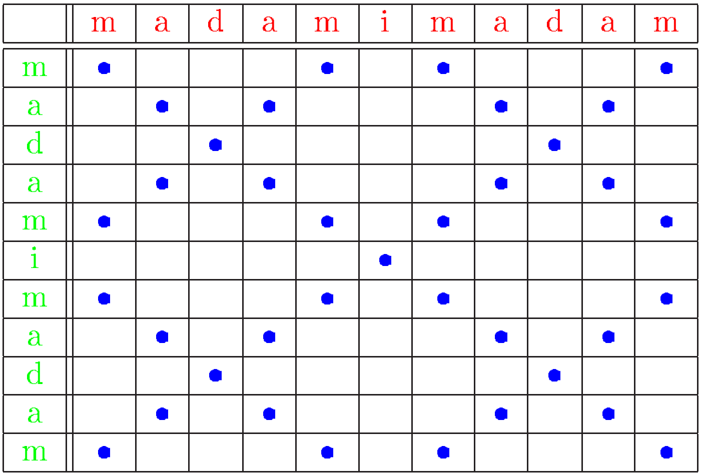
Notice two things about k-error matching. (i) Its result is a type of alignment we have not seen so far, as the query is aligned globally, while its match to the subject is local. Colloquially we call this a *glocal* alignment. (ii) For biological sequences the value of k compatible with homology is usually unknown. In spite of this, the strategy of `MUMmer` is clearly analogous. The central idea is to exclude "unpromising" regions from dynamic programming through the use of exact matching. This exclusion strategy also forms the basis of two of the most popular search tools for biological databases.

4.2.2 Examples of Database Search Programs

Currently, perhaps the most frequently applied database search programs are FASTA and BLAST. Both are based on the exclusion idea. FASTA was published in 1988 and quickly became widely used by molecular biologists [197]. BLAST was published two years later as an attempt to improve on the performance of FASTA [9]. Both programs have evolved considerably since their inception more than a decade ago. Here we restrict ourselves to outlining central algorithmic ideas incorporated in the original versions.

FASTA

A common method for visualizing similarities between two sequences is known as a *dotplot*. Two sequences are written along the two dimensions of a grid and matches are marked by dots. Figure 4.6 shows a dotplot comparing the string `madamimadam` with itself. As a consequence of the self-comparison, all cells on the diagonal from the top left to the bottom right contain a dot. Moreover, `madamimadam` is a palindrome and hence the opposite diagonal going from the bottom left to the top right



	m	a	d	a	m	i	m	a	d	a	m
m	●				●		●				●
a		●		●				●		●	
d			●						●		
a		●		●				●		●	
m	●				●		●				●
i						●					
m	●				●		●				●
a		●		●				●		●	
d			●						●		
a		●		●				●		●	
m	●				●		●				●

Fig. 4.6. Dotplot comparing the string `madamimadam` with itself. Each pair of identical characters is marked by a dot.

cell is also dotted[1]. In addition, the off-diagonal dots indicate the fact that `madam` is repeated as well as palindromic itself.

Dotplots of DNA sequences are usually heavily populated with dots. It is therefore customary to restrict them to runs of matches of a certain minimal length, which then appear as diagonals. An example of this is shown in Figure 4.7, where the sequences of the mRNAs of human δ-globin and γ-globin are compared. Figure 4.7C suggests that the central portions of both sequences are more highly homologous than the approximately 100 bp at their 5' and 200 bp at their 3' ends.

The central idea for dramatically speeding up local alignment compared to straight dynamic programming is to concentrate on diagonals with the highest score in order to exclude the other areas of the matrix from explicit searching [253, p. 172]. Notice that in the context of FASTA the dotplot is used only metaphorically to get an intuitive understanding of the algorithm; it is not a data structure actually built during execution of the algorithm.

Instead, the central data structure used by FASTA is known as a *hash table*. This is a generalization of an array. An array consists of entries that are indexed by monotonously increasing integers usually starting at 0 or 1. A hash table is an array that is indexed by arbitrary objects such as strings. This hash index is converted to an ordinary array index using a hash function [41, ch. 11]. The great advantage of such a data structure is that searching for one of the indices, also referred to as *keys*, is very quick. The hash table we are interested in has as keys words of a certain length and as entries their positions in the text. For example, if we hash the text $T =$ `ACCAGAGAATT` into words of length 2, we obtain Table 4.1.

Table 4.1. Text $T =$ `ACCAGAGAATT` hashed into words of length 2.

key	position
AC	1
CC	2
CA	3
AG	4, 6
GA	5, 7
AA	8
AT	9
TT	10

FASTA proceeds in four steps:

1. The query sequence (pattern) is hashed into words of length $ktup$. This hash table is then used to localize exact matches of substrings of length $ktup$ in the

---

[1] Notice that in molecular biology, as opposed to natural languages, a palindrome is a sequence that is equal to its reverse complement, for example `ACGT`.

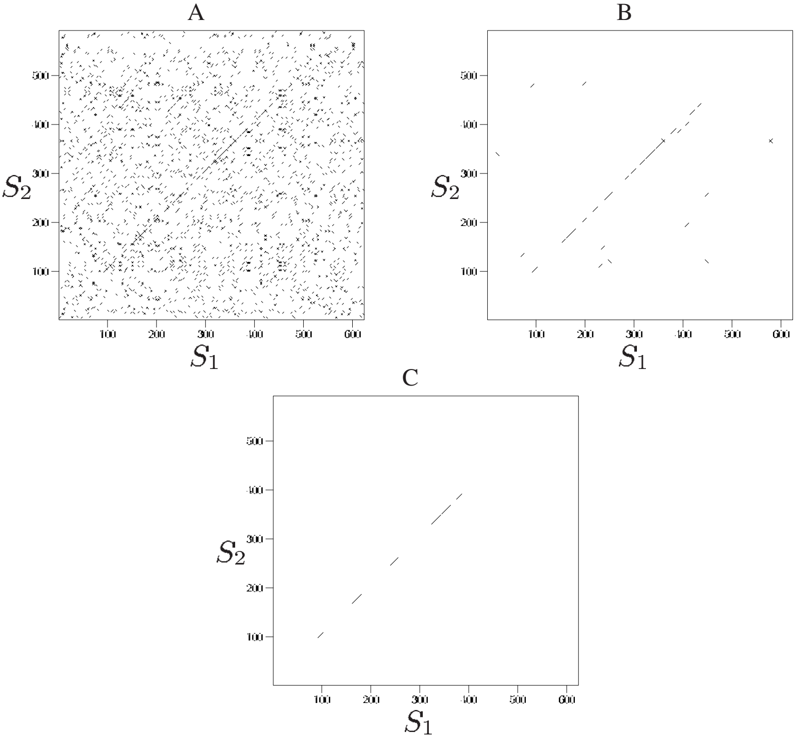


Fig. 4.7. Dotplot of the mRNA sequences of human δ-globin (S_1) and γ-globin (S_2). **A**: Matches of length ≥ 4; **B**: matches of length ≥ 8; **C**: matches of length ≥ 12. Notice the increasing sparseness of the plots, as well as the fact that, in contrast to the dotplot in Figure 4.6, the vertical sequence runs from bottom to top.

subject sequence (text). These matches can be imagined as diagonals of length $ktup$ in a dotplot.

2. The ten diagonals with the highest density of matches are rescored using a scoring matrix (in the case of protein sequences) or a simple DNA scoring scheme.
3. The diagonal regions that after rescoring have a score above some threshold are joined into one region.
4. The joined region is aligned by dynamic programming.

In order to gain some insight into the time requirement of this algorithm, let us concentrate on the filtering of exact matches from step 2 onwards. This takes time proportional to the number of dots in the dotplot [253, p. 176]. For random sequences of lengths m and n this amounts to

$$O\left(m \cdot n \cdot P(\mathrm{match})\right)^{ktup},$$

which is essentially $O(m \cdot n)$. This might strike the reader as a rather disappointing result, as it is identical to the time requirement of standard dynamic programming.

However, the probability of finding a match of length $ktup$, $P(\mathrm{match})^{ktup}$, is usually very small, thereby still speeding up execution considerably. Consider, for instance, an alphabet of equiprobable nucleotides and $ktup = 8$. In this case $P(\mathrm{match})^{ktup} = (1/4)^8 \approx 1.5 \cdot 10^{-5}$, amounting to a potential time saving of orders of magnitude [197].

As in the `MUMmer` program introduced in Section 4.1, FASTA proceeds by applying dynamic programming only to a very small portion of the space of possible alignments. The next logical step would be to try doing without dynamic programming altogether. This was also the conclusion of the developers of the **B**asic **L**ocal **A**lignment **S**earch **T**ool described next.

BLAST

BLAST was published two years after FASTA and at the time was an order of magnitude faster [9, 18]. Like FASTA, BLAST is based on the exclusion of unpromising areas of the alignment matrix from detailed searching. The central notion of the BLAST algorithm is the **m**aximal **s**egment **p**air (MSP). This is a pair of segments (substrings) of identical length whose score cannot be improved by extension on either side.

Figure 4.8 shows the three steps of BLAST: (i) compilation of a list of high-scoring words, (ii) a scan of the subject sequences contained in the database for exact matches to these words, and (iii) extension of the exact matches. We cover each step in turn.

Construction of a list of high-scoring words (Fig. 4.8A) is essentially a refinement of the $ktup$-word approach described in the context of FASTA. Given a protein query sequence, all substrings (words) of a certain length w that have a match with a score $\geq T$ somewhere in the query are stored along with their match positions in the query. Table 4.2 shows the word list constructed from the example query `AHYV` based on the BLOSUM62 score matrix and $w = 2$. This word list contains 15 entries, which is a considerable expansion of the three exact matches that can be formed (`AH`, `HY`, and `YV`). In practice, a word length of 3 is frequently used for protein sequences. In conjunction with the PAM120 substitution matrix, $T = 17$ was found to give satisfactory results [9]. The list of words [179] resulting from the choice of w and T contains approximately 50 entries for each residue in the query, such that a protein sequence of length 100 would induce a word list comprising perhaps 5,000 entries [9].

The preprocessing of a DNA query sequence is simpler than that of proteins just described. Given a DNA query, it is hashed into words of length w, which is typically set to 12.

The scanning phase (Fig. 4.8B) poses an exact matching problem that should by now look familiar to the reader. If the database is stable, its transformation into the suffix tree format would allow a rapid retrieval of all positions of word matches. However, the authors of BLAST chose the far more memory-efficient option of preprocessing the distinct query words into a keyword tree. The scan phase therefore

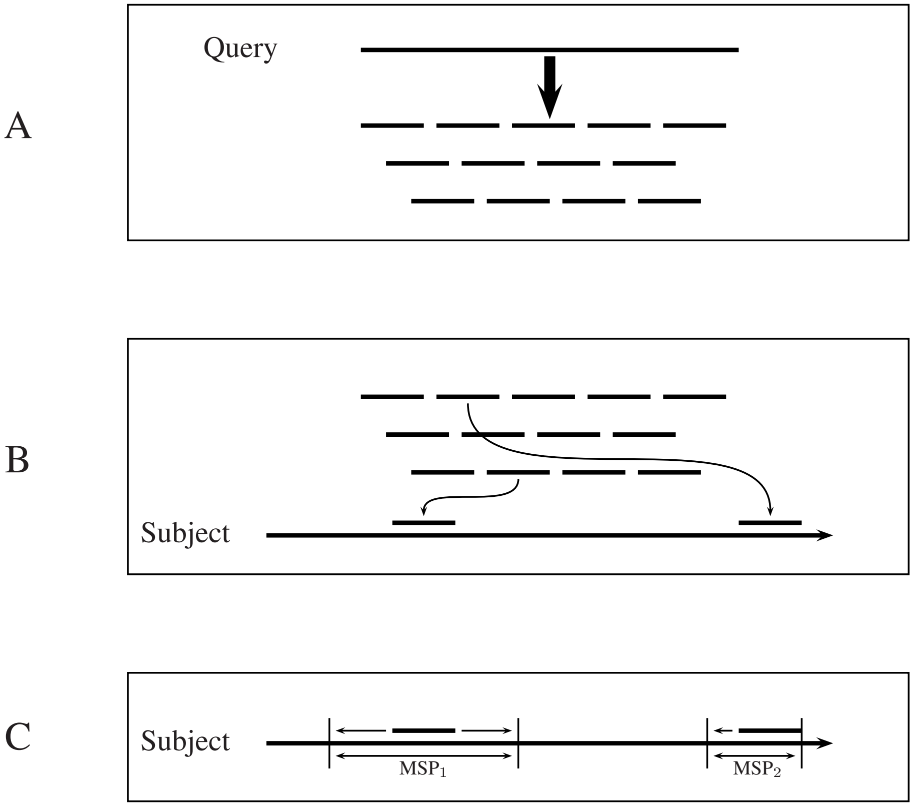


Fig. 4.8. The three steps in the BLAST algorithm. **A**: Generate list of high-scoring words from query; **B**: search for exact matches between members of the word list and subject; **C**: generate maximal segment pairs (MSPs) by extending exact matches to left and right as long as the score increases. Notice that MSP$_2$ could only be extended to the left.

takes $O(n + l + k) \approx O(n)$ time, where n is the size of the database, l the length of the word list, and k the number of matches of members of the word list.

Finally, in the extension phase (Fig. 4.8C) matches returned from the scanning phase are extended in either direction. This continues until the score of the MSP remains below a previous maximum score for, say, 20 residues. The extension phase takes time proportional to the number of matches returned by the scanning phase. Under a random sequence model this is proportional to the probability of finding a certain word of length w, $P(\text{match})^w$, multiplied by the size of the database and the length of the word list: $O(n \cdot l \cdot P(\text{match})^w)$. If we model our database as consisting of equiprobable nucleotides, this time requirement becomes $O(nl/4^w)$ [9].

Run time assessments are associated with a classical trade-off between *sensitivity* and *specificity* encountered in many database search programs. Sensitivity is the proportion of true homologues returned by a search. Consider, for example, a data-

Table 4.2. List of high-scoring words constructed from the query sequence `AHYV` using word length $w = 2$, score threshold $T = 8$, and the BLOSUM62 substitution matrix (Table 2.4).

Member of Word List	Match in Query	Score
AH	AH	12
CH	AH	8
GH	AH	8
SH	AH	9
TH	AH	8
VH	AH	8
HY	HY	15
NY	HY	8
YY	HY	9
HF	HY	11
HW	HY	10
YV	YV	11
YI	YV	10
YL	YV	8
YM	YV	8

base containing 20 MAP kinases. A search for MAP kinases returns 25 sequences of which 18 are MAP kinases. In this case the sensitivity of the search program is $18/20$. This quantity can trivially be maximized by returning the entire database. In contrast, specificity is the portion of true homologues among the set of hits returned. The specificity of our example search for MAP kinases is $18/25$. Again, this can be maximized in a trivial fashion by returning very few high-scoring hits. An ideal database search program is characterized by specificity $=$ sensitivity $= 1$.

In the case of BLAST, the algorithm runs in time proportional to the sum of the time taken by each of the three steps outlined above: $O(l+n+n\cdot l/4^w)$ [9]. The larger the word list (i.e. l), the smaller the probability that a significant MSP will be missed, i.e. the higher the sensitivity. On the down side, this leads to a longer run time and possibly a decrease in specificity. For a given query, the size of the word list is determined by the word length, w, and the threshold score, T. Increasing w inflates the word list, but this effect is offset by the reduced likelihood of finding the longer word in the database. This leaves T as the critical parameter determining speed and sensitivity: Lowering T increases the size of the word list thereby making the program more sensitive and also slower. However, there appear to be diminishing returns to lowering T, and in test runs a value of $T = 17$ in conjunction with a PAM120 score matrix recovered virtually all MSPs from various protein superfamilies [9].

4.3 Database Composition

In our discussion of the k-error string matching algorithm as well as of FASTA and BLAST we have repeatedly referred to a random sequence model. This simplified our time analyses, but biological sequences are highly non-random due to locally biased base composition (e.g. A/T-rich regions) and repetitive elements. A successful database search tool needs to take account of this fact, since a query sequence containing a region that matches a repeat element might otherwise return a huge number of spurious results. For this reason BLAST incorporates two somewhat *ad hoc* but effective filtering steps: The first scans the frequency of all 8-mer words in the database and ignores matches to words with grossly elevated frequencies. The second scans a compact library of repetitive elements and ignores query words that return a match to one of its members.

4.4 Heuristic *vs.* Optimal Alignment Methods

All three of the fast alignment programs mentioned in this chapter, `MUMmer`, FASTA, and BLAST, are based on so-called *heuristic* algorithms. In contrast, the alignment algorithms based exclusively on dynamic programming are known as optimal methods as they guarantee to yield the best alignment under the scoring scheme. In the context of `MUMmer` the heuristic nature of the algorithm means that there might be alignments of the input sequences that have a higher score than the alignment returned by the program. Similarly, for heuristic local alignment programs, there is a (usually small) probability that a significant match does not contain a $ktup$ word or a member of the word list employed by BLAST and the corresponding homology is missed. However, extensive tests have shown that in practice this problem is negligible compared to the huge saving in time and memory achieved by the three programs.

Database searching raises not only algorithmic issues but also the statistical problem of deciding whether a given local alignment might be due to chance alone. Perhaps the most important innovation of BLAST was to incorporate statistical theory allowing the direct computation of the significance of an MSP [134].

4.5 Application: Determining Gene Families

Many real-world applications of database search programs are critically dependent on an assessment of the significance of a given alignment. We introduce such an application in the context of the investigation of gene families in complex genomes.

In many eucaryotic organisms the deletion of a gene (knockout) has no phenotypic effect. On the other hand, the genome projects of the past decade have confirmed that in eucaryotes many genes exist as members of gene families. In fact, the probability of a gene being duplicated has been estimated to be as high as 1% per million years, which is greater than the probability that the gene mutates in this time window [170]. This high incidence of gene duplication suggests that the lack of phenotypic effects in knockouts might in many cases be due to compensation by a duplicated gene.

In order to investigate this idea further, Li and coworkers took advantage of two extraordinary data sets: (i) the complete genome sequence of brewer's yeast, *Saccharomyces cerevisiae*, which contains 6,357 genes and (ii) a set of measurements on the fitness effects of knockouts for 5,766 of these genes [93]. The first step in the analysis was to distinguish genes that were not duplicated ("singletons") from duplicated ones. This was done by an all-against-all FASTA search of the 5,766 protein sequences of the genes included in the study. Singletons were defined as genes that had no significant homology to any other gene in the genome, while duplicates had at least one significant hit and in addition could be aligned over $\geq 50\%$ of their sequence with their putative homologue.

Figure 4.9 illustrates a procedure for extracting gene families from these pairwise comparisons. The results of the all-against-all comparison of a proteome just outlined can be summarized in a pairwise match matrix $M = \{m_{ij}\}$. A match between protein p_i and p_j is denoted by $m_{ij} = 1$ and $m_{ij} = 0$ if no homology was detected or if $i = j$. The protein families are now assembled by starting from the first entry equal to 1 in the matrix and recursively visiting the next 1 in that column followed by the next 1 in that row. Every time a 1 is encountered, the corresponding protein is added to the current cluster and the matrix entry is set to zero. This procedure is repeated until all entries in M are zero. An analogous but more sophisticated clustering strategy is implemented in the program `blastclust`, which is distributed as part of the BLAST software package.

Proteome Match Matrix Protein Families

$$\{p_1, p_2, p_3, p_4, p_5\} \longrightarrow \begin{array}{|c|c|c|c|c|c|}\hline & p_1 & p_2 & p_3 & p_4 & p_5 \\ \hline p_1 & 0 & 1 & 0 & 0 & 0 \\ \hline p_2 & 1 & 0 & 0 & 1 & 0 \\ \hline p_3 & 0 & 0 & 0 & 0 & 1 \\ \hline p_4 & 0 & 1 & 0 & 0 & 0 \\ \hline p_5 & 0 & 0 & 1 & 0 & 0 \\ \hline \end{array} \longrightarrow f_1 = \{p_3, p_5\},\ f_2 = \{p_1, p_2, p_4\}$$

Fig. 4.9. Strategy for the detection of protein families within a proteome.

Such a procedure led to the identification of 1,275 singletons and 1,147 gene families in the yeast proteome. In the subsequent comparison between the singleton/duplicate classification and the data set on fitness effects of knockouts, duplicated genes had a significantly higher probability of functional compensation than singletons. Overall approximately 1/4 of deletions with no phenotypic effect could be accounted for by gene duplication [93].

4.6 Statistics of Local Alignments

In Section 4.4 the notion of a *significant* homology was central to the classification of genes into families. The direct calculation of this significance was one of the major advances in the development of fast local alignment tools.

4.6.1 Maximum Local Alignment Scores

The distribution of extreme values taken from multiple samples, be it the maximum height of people in various towns, the maximum length of proteins in a sample of proteomes, or the maximum score of local alignments between random sequences, are described by the extreme value distribution. Its probability density function is given by

$$P(x) = \lambda e^{(\mu - x)\lambda - e^{(\mu - x)\lambda}}, \tag{4.2}$$

where μ is the location parameter and λ the dispersion parameter. Figure 4.10 shows this probability density function for $\mu = 0$ and $\lambda = 1$. The cumulative distribution function of the extreme value distribution is

$$C(x) = e^{-e^{(\mu - x)\lambda}}, \tag{4.3}$$

which is also shown in Figure 4.10 for $\mu = 0$ and $\lambda = 1$.

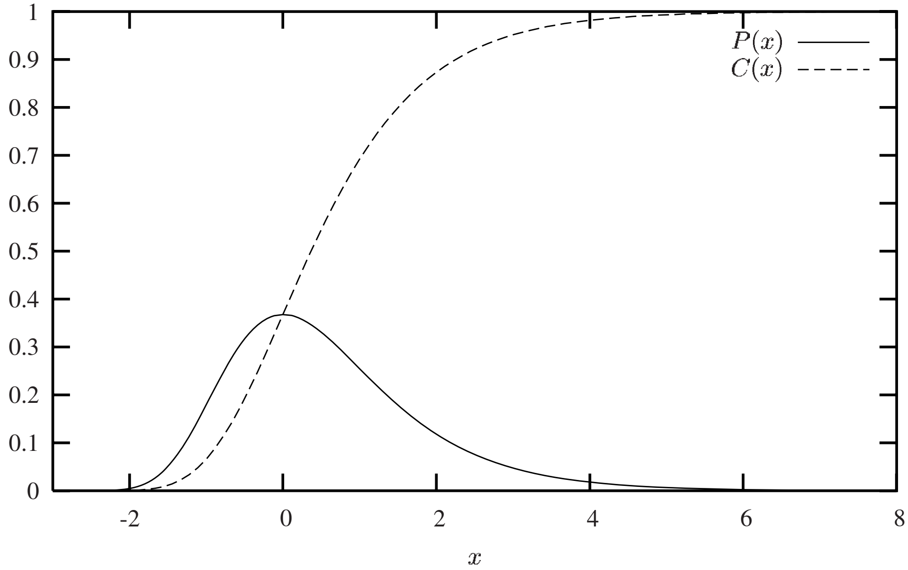


Fig. 4.10. Probability density function, $P(x)$, and cumulative distribution function, $C(x)$, of the extreme value distribution with parameters $\mu = 0$ and $\lambda = 1$.

In order to estimate the parameter λ for the distribution of maximum local alignment scores, let p_i be the background frequency of the i-th residue in the database, e.g. the amino acid background frequencies displayed in Figure 2.7. Then λ is the unique positive solution of

$$1 = \sum_{i,j} p_i p_j e^{\lambda s_{ij}},$$

where s_{ij} is the log-odds score of the residue pair $p_i p_j$ taken from a given substitution matrix, e.g. BLOSUM62 (Table 2.4) [134]. The parameter μ is estimated as

$$\mu = \frac{\ln(Kmn)}{\lambda}, \tag{4.4}$$

where m and n are the *effective* lengths of the query and subject sequences, and K is a scaling parameter. The effective lengths of query and subject are less than their raw lengths, which is due to the fact that high-scoring local alignments cannot start at the ends of sequences. This edge effect, that is the difference between effective and actual lengths, tends to become negligible for sequences longer than 200 residues. K, which lies between 0 and 1, corrects for the fact that the $m \cdot n$ entries in an alignment matrix are not independent, as the paths to which these scores refer may intersect. Computation of K is more involved than λ and the interested reader is referred to a publication by Karlin and Alschul [134]. By substituting into the complement of Equation (4.3), we obtain the probability of observing a score greater or equal to some threshold score by chance alone:

$$P(S \geq x) = 1 - e^{-Kmne^{-\lambda x}}. \tag{4.5}$$

As an example application of the statistics of local alignments, we compare human β-globin with a lupine leghemoglobin, which consist of 146 and 153 amino acids, respectively. Using the BLOSUM62 substitution matrix, a gap opening score of -12 and a gap extension score of -1, the optimal alignment is shown in Figure 4.11 and has a score of 39. Is this score significant?

```
human β-globin:       50 TPDAVMGNPKVKAHGKKV 67
                         | .    ||...||  ||
lupine leghemoglobin: 50 TSEVPQNNPELQAHAGKV 67
```

Fig. 4.11. Optimal local alignment between human β-globin and lupine leghemoglobin.

According to the `BLAST` output, the effective length of β-globin is 130, that of leghemoglobin is 137, $\lambda = 0.267$, and $K = 0.041$. Hence we can compute $\mu = \ln(130 \cdot 137 \cdot 0.041)/0.267 \approx 24.69$. By substituting these estimates for μ and λ into Equation (4.3) and taking the complement, we get $P(S \geq x) = 0.022$.

Figure 4.12 shows the distribution of 100,000 maximum local alignment scores obtained by shuffling β-globin and realigning it to leghemoglobin. This can be compared to the expected distribution obtained by substituting $\mu = 24.69$ and $\lambda = 0.267$

into the probability density function given by Equation (4.2). The simulated probability of obtaining a score of ≥ 39 by chance alone is 0.024, which is very close to the result of 0.022 obtained much more efficiently by applying the extreme value distribution. We can conclude that the alignment found is marginally significant.

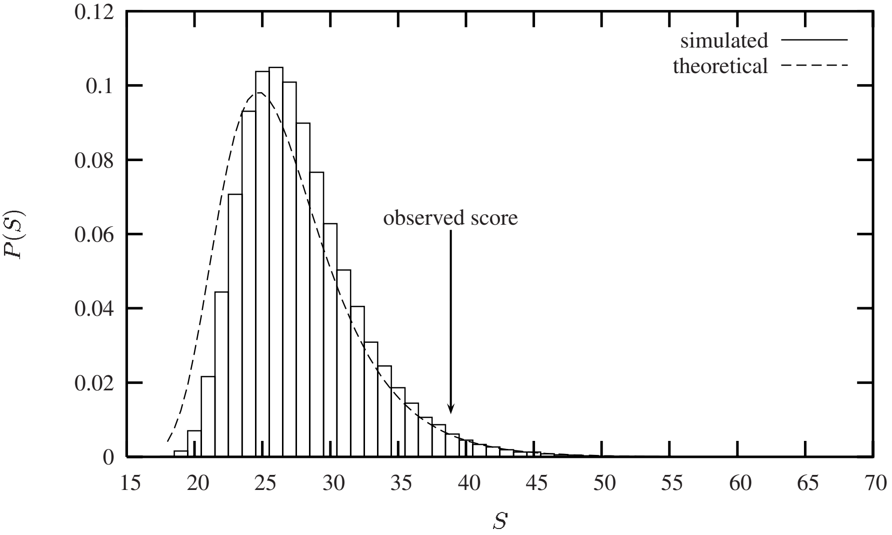


Fig. 4.12. Simulated and theoretical distribution of optimal local alignments scores. The simulated distribution was obtained by shuffling human β-globin 100,000 times and aligning it to lupine leghemoglobin.

Instead of the probability of getting a score greater or equal to that observed, $P(S \geq x)$, some implementations of `BLAST` quote the expected number of alignments with a score of at least x. This is known as the E-value, which is calculated by assuming that a given dynamic programming matrix contains only few entries with a score $S \geq x$. The distribution of rare events can often be modeled by the Poisson distribution. We start by considering the probability of an event occurring at least once, which is simply the complement of the event not occurring at all. In the context of our discussion, the probability of finding an alignment with score $S \geq x$ is the complement of finding no significant alignment: $P(S \geq x) = 1 - P(k = 0)$. For Poisson-distributed events $P(k = 0) = e^{-\mu}$, where μ is the mean of the distribution. Here $\mu = E(S \geq x)$, and we solve

$$P(k = 0) = e^{-E(S \geq x)}$$

for $E(S \geq x)$ to obtain

$$E(S \geq x) = -\ln(P(k = 0)).$$

Hence we get the expectation value by taking the logarithm of the complement of Equation (4.5) and multiplying by -1:

$$E(S \geq x) = Kmne^{-\lambda x}. \tag{4.6}$$

It is important to remember the distinction between the expectation value attached to a score and its significance. The range of expectation values (E) is bounded by zero and Kmn, while probabilities lie between zero and one. However, for small probabilities, say $P < 0.01$, the E-value and the corresponding P-value are very similar as shown in Figure 4.13.

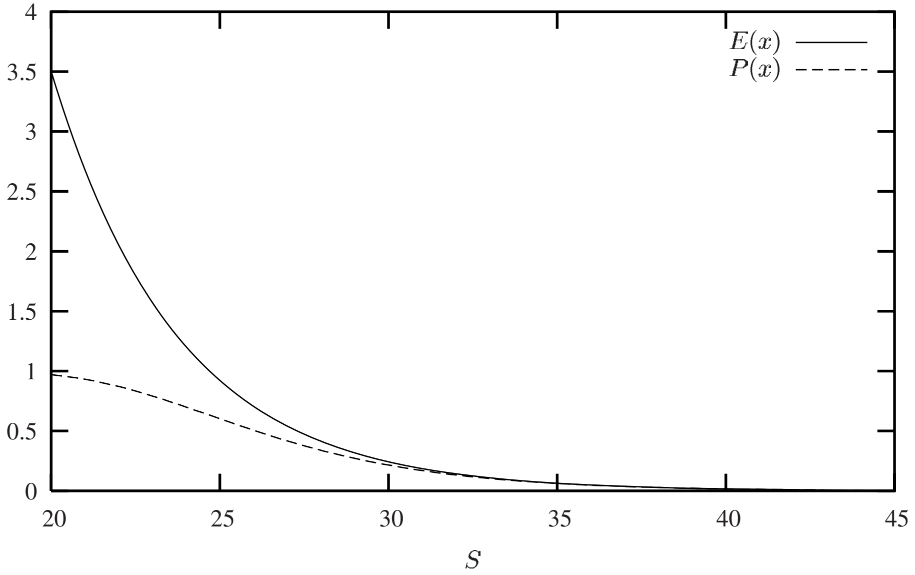


Fig. 4.13. The number of random alignments expected to have a score $\geq x$, $E(S \geq x)$, and the probability of obtaining a score $\geq x$, $P(S \geq x)$, as a function of the score, S. The parameters for this example are taken from the alignment of human β-globin and lupine leghemoglobin described in the text, in which case $Kmn = 730.21$ and $\lambda = 0.267$.

4.6.2 Choosing a Substitution Matrix

We explained in Section 2.10 that the mean score of a random pair of residues should be negative for local alignments. However, the difference in score between a random and a statistically significant match is maximized if the entries in the scoring matrix fulfill a further requirement. Let p_i be the frequency of amino acid i in the database and let q_{ij} be the probability with which amino acids i and j are found in homologous positions in protein alignments. Then the scoring scheme that best distinguishes significant from spurious alignments contains entries for residues i and j of the form [8]

$$s_{ij} = \ln\left(\frac{q_{ij}}{p_i p_j}\right).$$

This is the log-odds form of the entries found in both the PAM and the BLOSUM matrices (cf. Section 2.4). Moreover, entries of this format are on average negative, i.e.

$$\sum p_i p_j s_{ij} < 0.$$

Of the two quantities that go into calculating an entry in a scoring matrix, q_{ij} is the more interesting. It represents the frequencies of homologous pairs of residues. This set of "target frequencies" changes over evolutionary time, hence the series of scoring matrices. By choosing one of these, the user essentially chooses the set of target frequencies she judges appropriate for the alignment problem in hand. Notice that if evolution has gone on for long enough as to completely randomize residues in homologous positions, $q_{ij} = p_i p_j$. In other words, in the limit of infinite time the target frequencies become equal to the corresponding product of the background frequencies and all log-odds are zero.

4.7 Bit Scores

So far we have considered "raw scores". These have the disadvantage of being dependent on K and λ, i.e. on parameters that are functions of the score matrix employed and the background frequencies of the residues in the database. This makes it difficult to compare the significance of alignments obtained using different score matrices. In order to obtain expectation values that are independent of the score matrix, a bit score S' has been defined:

$$S' = \frac{\lambda S - \ln K}{\ln 2}. \tag{4.7}$$

Substituting Equation (4.7) into Equation (4.6) we get

$$E(S' \geq x) = mn2^{-x}. \tag{4.8}$$

The corresponding significance is obtained by the Poisson approximation already outlined:

$$P(S' \geq x) = 1 - e^{-E(S' \geq x)}.$$

4.8 Summary

The comparison of DNA and protein sequences is one of the central applications of bioinformatics in biological research. As the amount of data available for meaningful comparisons has grown exponentially since the early 1980s, rapid algorithms for approximating alignments obtained by optimal methods have become central to the practice of sequence analysis. The methods for rapid global and local alignment introduced in this chapter are all based on the idea of excluding large parts of the sequences compared from detailed investigation. In the global approach, the aim is to cover as much of the alignment as possible through exact matches between the two sequences analyzed. These exact matches can be discovered efficiently by the application of a generalized suffix tree. In the local approach, promising regions in the database are identified through an exact match to a substring of the query. This procedure is heuristic as there is a (small) probability that a homologous sequence might not contain an exact match to the substrings of the query. Designing the set of query substrings is therefore a critical step in database search programs such as FASTA and BLAST. Since these programs often scan large databases, it is important to gain a measure of statistical significance for a given alignment. The statistical theory accounting for the significance of local alignments is formally restricted to ungapped alignments, but appears to work equally well for gapped alignments [198]. The significance of an alignment is summarized by the expectation value shown in Equation (4.8). This corresponds to the number of alignments with a score greater or equal to the one observed that are expected to occur by chance alone.

4.9 Further Reading

Bedell and his colleagues have written a comprehensive guide to the BLAST software package [18] .

4.10 Exercises and Software Demonstrations

4.1. Consider a set of sequences containing 40 hexokinases. You compare a known hexokinase to your collection using a pairwise alignment tool and obtain 35 hits, of which 28 are true hexokinases. What is the sensitivity and specificity of your search program?

4.2. Consider the following sequence of numbers

$$2, 1, 5, 7, 3, 4, 2, 7, 2, 1$$

and use dynamic programming to find its longest increasing subsequence.

4.3. The genome of *E. coli* K12 has the following composition:

`A` 1,142,228 bp
`C` 1,179,554 bp
`G` 1,176,923 bp
`T` 1,140,970 bp

1. What is the probability of drawing two identical nucleotides from this genome?
2. What is the expected length of the longest repeat sequence?

3. Let P be an *E. coli* query sequence of length 100 nucleotides. In a search with $k = 9$ errors, what is the length of the substring of P that is guaranteed to be error-free?
4. How many exact matches of this length are expected to occur in the *E. coli* genome?

4.4. Hash the string `AGTAAC` into substrings of length 2 and compare your result to that obtained using the program `Match`→`Hash Table` in the `bioinformer` software (Section A.2.4).

4.5. Familiarize yourself with the idea of a dotplot by working through the tutorial for the `bioinformer` program `Match`→`Dotplot` in Section A.2.5).

4.6. How would you construct an alternative solution to Exercise 3.2 that runs in time linear in the sum of the lengths of an arbitrary number of input strings?

4.7. You are given an unknown human protein and compare it to the mouse proteome using BLAST. An abridged version of the BLAST output is shown in Figure 4.14.

1. What is the function of the unknown protein?
2. Is `EWL` part of the query sequence's word neighborhood?
3. Calculate the raw score of the local alignment between the query and the sushi-repeat containing protein.
4. Convert the raw score just computed to a bit score and compare your result to the corresponding bit score on the printout.
5. Is the homology to the sushi-repeat containing protein reported in Figure 4.14 significant?
6. You repeat the BLAST search a year later using the same database, whose size has doubled in the meantime. Everything else being equal, what result do you expect for
   a) the score?
   b) the E-value?

4.8. Given the BLOSUM62 score matrix, the raw score of the optimal local alignment between human β-globin and lupine leghemoglobin shown in Figure 4.11 is 39 and its bit score is 19.6.

1. Recalculate the raw score based on the BLOSUM45 matrix shown in Table 4.3.
2. Convert the raw score to a bit score using the parameters $\lambda = 0.195$ and $K = 0.032$ returned by BLAST for the alignment of β-globin and leghemoglobin under the BLOSUM45 substitution matrix.

```
                                                              Score    E
Sequences producing significant alignments:                   (bits) Value

gi|NP_062728.1| neuronal protein; neuronal protein N...        408   e-114
gi|NP_848713.1| transgelin 2 [Mus musculus]                    308   2e-84
gi|NP_081114.1| sushi-repeat containing protein [Mu...          27   9.9

>gi|NP_062728.1| neuronal protein; neuronal protein Np25
           [Mus musculus]
          Length = 199

 Score =  408 bits (1048), Expect = e-114
 Identities = 198/199 (99%), Positives = 198/199 (99%)

Query: 1   MANRGPSYGLSREVQEKIEQKYDADLENKLVDWIILQCAEDIEHPPPGRAHFQKWLMDGT 60
           MANRGPSYGLSREVQEKIEQKYDADLENKLVDWIILQCAEDIEHPPPGRAHFQKWLMDGT
Sbjct: 1   MANRGPSYGLSREVQEKIEQKYDADLENKLVDWIILQCAEDIEHPPPGRAHFQKWLMDGT 60


>gi|NP_848713.1| transgelin 2 [Mus musculus]
          Length = 199

 Score =  308 bits (790), Expect = 2e-84
 Identities = 144/199 (72%), Positives = 170/199 (85%)

Query: 1   MANRGPSYGLSREVQEKIEQKYDADLENKLVDWIILQCAEDIEHPPPGRAHFQKWLMDGT 60
           MANRGPSYGLSREVQ+KIE++YDADLE  L+ WI  QC ED+   P PGR +FQKWL DGT
Sbjct: 1   MANRGPSYGLSREVQQKIEKQYDADLEQILIQWITTQCREDVGQPQPGRENFQKWLKDGT 60


>gi|NP_081114.1| sushi-repeat containing protein [Mus
           musculus]
          Length = 467

 Score = 27.3 bits (59), Expect = 9.9
 Identities = 12/44 (27%), Positives = 22/44 (50%)

Query: 226 HLLTFSSFSKPSVPGFCKCCISAENPRCLLLPPPVHLELCKDSA 269
           H++ ++++ +      CK  +  +  RC +L PP H  L   SA
Sbjct: 239 HVIRYTAYDRAYNRASCKFIVKVQVRRCPILKPPQHGYLTCSSA 282


  Database: mouse_protein.fa
    Posted date:  Dec 19, 2003  3:47 PM
  Number of letters in database: 11,582,853
  Number of sequences in database:  25,371

Lambda     K      H
   0.319    0.136    0.413

Gapped
Lambda     K      H
   0.267   0.0410    0.140


Matrix: BLOSUM62
Gap Penalties: Existence: 11, Extension: 1
```

Fig. 4.14. Abridged output of a BLAST search.

Table 4.3. BLOSUM45 amino acid substitution matrix. Match scores are shown in bold.

	A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W	Y	V
A	**5**	-2	-1	-2	-1	-1	-1	0	-2	-1	-1	-1	-1	-2	-1	1	0	-2	-2	0
R	-2	**7**	0	-1	-3	1	0	-2	0	-3	-2	3	-1	-2	-2	-1	-1	-2	-1	-2
N	-1	0	**6**	2	-2	0	0	0	1	-2	-3	0	-2	-2	-2	1	0	-4	-2	-3
D	-2	-1	2	**7**	-3	0	2	-1	0	-4	-3	0	-3	-4	-1	0	-1	-4	-2	-3
C	-1	-3	-2	-3	**12**	-3	-3	-3	-3	-3	-2	-3	-2	-2	-4	-1	-1	-5	-3	-1
Q	-1	1	0	0	-3	**6**	2	-2	1	-2	-2	1	0	-4	-1	0	-1	-2	-1	-3
E	-1	0	0	2	-3	2	**6**	-2	0	-3	-2	1	-2	-3	0	0	-1	-3	-2	-3
G	0	-2	0	-1	-3	-2	-2	**7**	-2	-4	-3	-2	-2	-3	-2	0	-2	-2	-3	-3
H	-2	0	1	0	-3	1	0	-2	**10**	-3	-2	-1	0	-2	-2	-1	-2	-3	2	-3
I	-1	-3	-2	-4	-3	-2	-3	-4	-3	**5**	2	-3	2	0	-2	-2	-1	-2	0	3
L	-1	-2	-3	-3	-2	-2	-2	-3	-2	2	**5**	-3	2	1	-3	-3	-1	-2	0	1
K	-1	3	0	0	-3	1	1	-2	-1	-3	-3	**5**	-1	-3	-1	-1	-1	-2	-1	-2
M	-1	-1	-2	-3	-2	0	-2	-2	0	2	2	-1	**6**	0	-2	-2	-1	-2	0	1
F	-2	-2	-2	-4	-2	-4	-3	-3	-2	0	1	-3	0	**8**	-3	-2	-1	1	3	0
P	-1	-2	-2	-1	-4	-1	0	-2	-2	-2	-3	-1	-2	-3	**9**	-1	-1	-3	-3	-3
S	1	-1	1	0	-1	0	0	0	-1	-2	-3	-1	-2	-2	-1	**4**	2	-4	-2	-1
T	0	-1	0	-1	-1	-1	-1	-2	-2	-1	-1	-1	-1	-1	-1	2	**5**	-3	-1	0
W	-2	-2	-4	-4	-5	-2	-3	-2	-3	-2	-2	-2	-2	1	-3	-4	-3	**15**	3	-3
Y	-2	-1	-2	-2	-3	-1	-2	-3	2	0	0	-1	0	3	-3	-2	-1	3	**8**	-1
V	0	-2	-3	-3	-1	-3	-3	-3	-3	3	1	-2	1	0	-3	-1	0	-3	-1	**5**

5

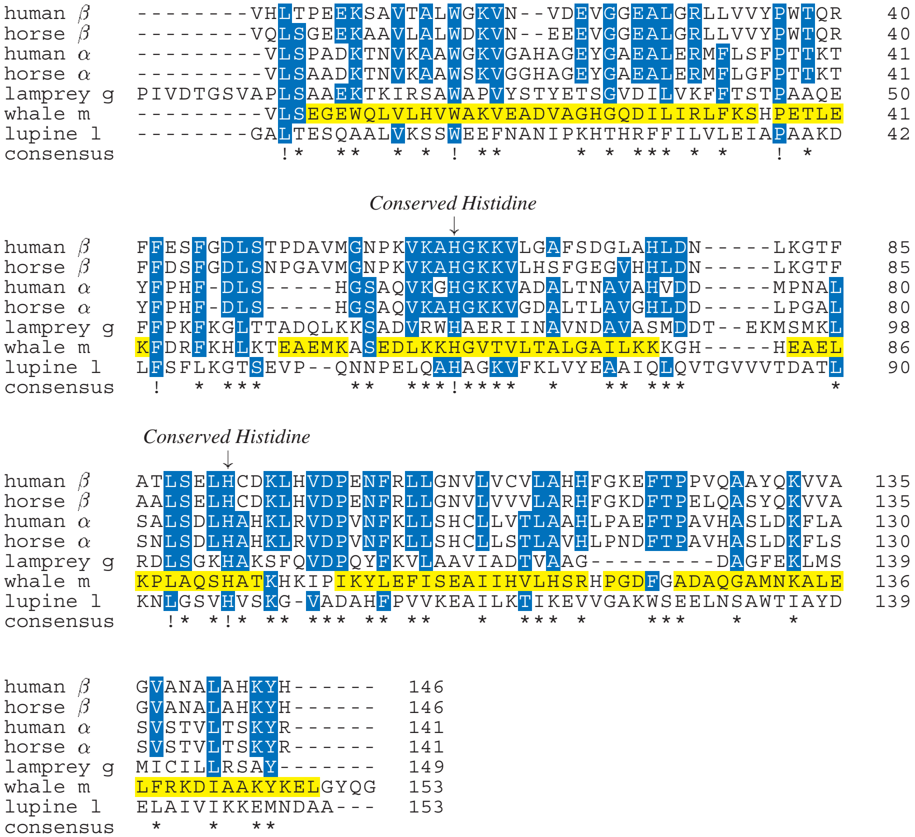
Multiple Sequence Alignment

> *The camel and the Llama are closely related species with different habitats. Camels live in the plains and Llamas high up in the Andes. The camel possesses a hemoglobin molecule with an affinity for oxygen that is normal for an animal of its size. But because of a single mutation in the gene coding for one of the two globin chains that make up a hemoglobin molecule, the Llama has a hemoglobin with an unusually high oxygen affinity. The variant hemoglobin helps the Llama breathe in the rarefied mountain air.*
>
> *Max Perutz [200, p. 204]*

The motivation for carrying out pairwise alignment is often to infer function from sequence similarity. In the case of multiple alignments this motivation is reversed: Given that a group of protein sequences have similar functions, we would like to know which sequence features are essential for this function. Such essential sequence motifs should be conserved, i.e. be similar, between homologous sequences from diverse species. The easiest way to discover conservation is to align many sequences encoding the function of interest and to look for invariant regions.

Figure 5.1 shows an alignment of seven globins, which are heme-containing proteins involved in oxygen storage or oxygen transport [49]. The alignment contains four globin sequences from mammals, one globin sequence from a fish, one myoglobin sequence from a mammal and finally a globin from a leguminous plant, a leghemoglobin. The last common ancestor of plants and animals existed over one and a half billion years ago. In spite of this, the sequences can be aligned in a meaningful way using the software `clustalw` [246], perhaps the best-known programs available for this purpose.

How do we know that the alignment shown in Figure 5.1 is meaningful? One method is to compare it to the three-dimensional structure of one of the aligned proteins. Myoglobin from sperm whale (Fig. 5.1) was the very first protein whose structure was solved. This work was carried out by John Kendrew and published in 1960 [137], seven years after the structure of DNA had been determined in the same Cambridge laboratory [256]. Not only was Kendrew the first to solve the three-dimensional structure of a protein, he was also a proto-bioinformatician, for he extensively used the first computers acquired by Cambridge University after the second World War to compute the structure of myoglobin from X-ray diffraction data [47, p. 111ff]. A refined version of the original myoglobin model is shown in Figure 5.2. The figure displays the eight α helices that are also marked in the multiple alignment



```
human β     --------VHLTPEEKSAVTALWGKVN--VDEVGGEALGRLLVVYPWTQR   40
horse β     --------VQLSGEEKAAVLALWDKVN--EEEVGGEALGRLLVVYPWTQR   40
human α     ---------VLSPADKTNVKAAWGKVGAHAGEYGAEALERMFLSFPTTKT   41
horse α     ---------VLSAADKTNVKAAWSKVGGHAGEYGAEALERMFLGFPTTKT   41
lamprey g   PIVDTGSVAPLSAAEKTKIRSAWAPVYSTYETSGVDILVKFFTSTPAAQE   50
whale m     ---------VLSEGEWQLVLHVWAKVEADVAGHGQDILIRLFKSHPETLE   41
lupine l    --------GALTESQAALVKSSWEEFNANIPKHTHRFFILVLEIAPAAKD   42
consensus             !*  **  * * ! **     * * *** * *   ! *

                         Conserved Histidine
                                  ↓
human β     FFESFGDLSTPDAVMGNPKVKAHGKKVLGAFSDGLAHLDN-----LKGTF   85
horse β     FFDSFGDLSNPGAVMGNPKVKAHGKKVLHSFGEGVHHLDN-----LKGTF   85
human α     YFPHF-DLS-----HGSAQVKGHGKKVADALTNAVAHVDD-----MPNAL   80
horse α     YFPHF-DLS-----HGSAQVKAHGKKVGDALTLAVGHLDD-----LPGAL   80
lamprey g   FFPKFKGLTTADQLKKSADVRWHAERIINAVNDAVASMDDT--EKMSMKL   98
whale m     KFDRFKHLKTEAEMKASEDLKKHGVTVLTALGAILKKKGH-----HEAEL   86
lupine l    LFSFLKGTSEVP--QNNPELQAHAGKVFKLVYEAAIQLQVTGVVVTDATL   90
consensus    !  * ***      **  ***!****  *   **  ***          *

            Conserved Histidine
                  ↓
human β     ATLSELHCDKLHVDPENFRLLGNVLVCVLAHHFGKEFTPPVQAAYQKVVA  135
horse β     AALSELHCDKLHVDPENFRLLGNVLVVVLARHFGKDFTPELQASYQKVVA  135
human α     SALSDLHAHKLRVDPVNFKLLSHCLLVTLAAHLPAEFTPAVHASLDKFLA  130
horse α     SNLSDLHAHKLRVDPVNFKLLSHCLLSTLAVHLPNDFTPAVHASLDKFLS  130
lamprey g   RDLSGKHAKSFQVDPQYFKVLAAVIADTVAAG---------DAGFEKLMS  139
whale m     KPLAQSHATKHKIPIKYLEFISEAIIHVLHSRHPGDFGADAQGAMNKALE  136
lupine l    KNLGSVHVSKG-VADAHFPVVKEAILKTIKEVVGAKWSEELNSAWTIAYD  139
consensus     !* *!* ** *** ** **   *  *** *    ***   *   *


human β     GVANALAHKYH------   146
horse β     GVANALAHKYH------   146
human α     SVSTVLTSKYR------   141
horse α     SVSTVLTSKYR------   141
lamprey g   MICILLRSAY-------   149
whale m     LFRKDIAAKYKELGYQG   153
lupine l    ELAIVIKKEMNDAA---   153
consensus    *   *  **
```

Fig. 5.1. Alignment of seven globin sequences. Dark gray: identical amino acids; light gray: α helices in myoglobin; α/β: α/β-globin; g: globin 5; m: myoglobin; l: leghemoglobin.

(Fig. 5.1). In the alignment we also notice that there are seven amino acids that have not changed over their entire evolutionary history exceeding one and a half billion years. Two of these conserved positions are specially marked histidines, which interact with the heme group of the globins. The heme group is responsible for the ability of globins to act as oxygen transporters and the amino acid residues that interact most intimately with this molecule are necessary for the proper functioning of all globins. The two histidines and the heme molecule of myoglobin are displayed in Figure 5.2. From the multiple alignment we can infer the histidines in the homologous proteins that interact with the heme molecule.

Multiple alignments are ubiquitous in molecular biology. They are used to

1. find sequence motifs that characterize protein families;
2. discover homologies between an unknown sequence and known sequence families;
3. support the prediction of secondary and tertiary structure;
4. support the design of PCR primers;

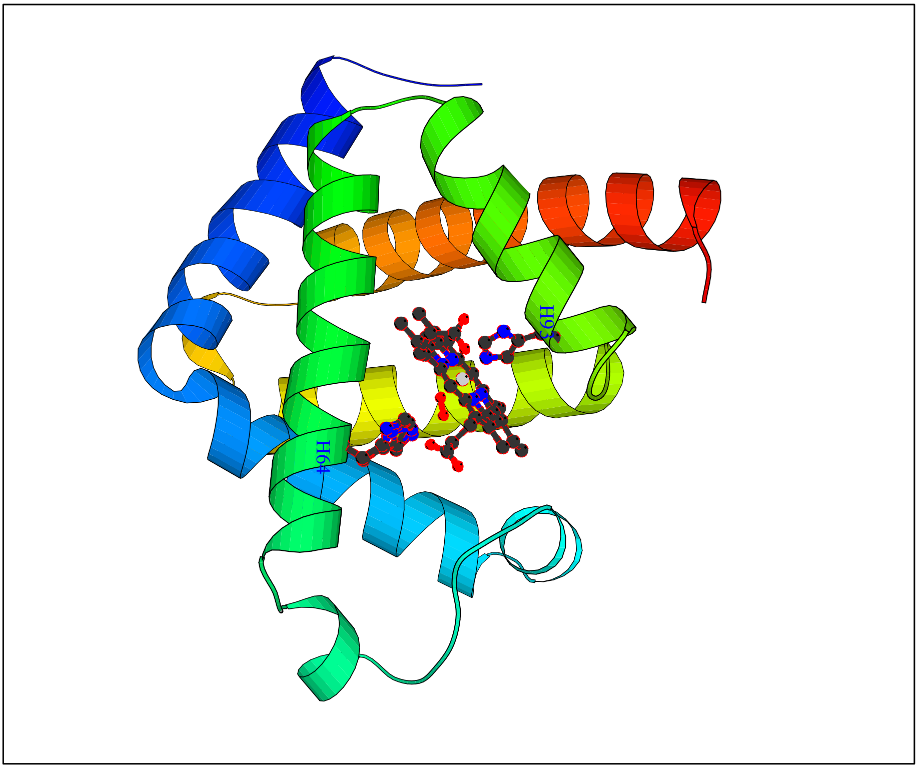


Fig. 5.2. 3D-Structure of oxy-myoglobin. The two histidines interacting with the heme group are shown in atomic detail. Notice also the bound oxygen molecule.

5. calculate phylogenetic trees.

In this chapter we will first look at the computation of multiple sequence alignments. We start by a natural extension of the dynamic programming algorithm explained in Section 2.6 and learn about a powerful speedup of this method. However, the speedup is not decisive enough to make dynamic programming a viable option when trying to align tens or hundreds of sequences, as one may wish to do when investigating all the members of a large protein family such as the globins [133]. In these cases heuristic multiple alignment methods developed since the 1980s are usually applied. In essence these methods reduce the multiple alignment problem to a series of pairwise alignments.

Heuristic multiple sequence alignments make use of two data structures not mentioned so far: sequence profiles and phylogenies. Both are used to compute multiple sequence alignments; on the other hand, they constitute classical applications of multiple sequence alignments. We treat phylogeny reconstruction in Chapter 8 and profile analysis in Section 6.1.

In the following discussion of the computation of multiple sequence alignments we concentrate on *global* approaches throughout, but local versions are of course also possible and have been implemented [229, 184].

5.1 Scoring Multiple Alignments

Before we can compute a multiple alignment we need a scoring scheme. One of the most popular schemes for scoring multiple sequence alignments is the sum-of-pairs method. Here the score of a multiple sequence alignment, M, is defined as

$$M = \sum_{i=1}^{|A|} \sum_{j}^{n} \sum_{k<j} s(S_j[i], S_k[i]),$$

where $|A|$ is the length of the alignment, n the number of sequences, and $s(S_j[i], S_k[i])$ is the score of position i of sequence j aligned with position i of sequence k. The alignment of two gap symbols is scored as zero.

5.2 Multiple Alignment by Dynamic Programming

As explained in Section 2.6, dynamic programming based on a two-dimensional matrix can be used to efficiently compute the score of a pairwise alignment. For n sequences the corresponding n-dimensional matrix can be represented as an n-dimensional hyperlattice. Figure 5.3 displays the cuboid for aligning three sequences. The path through this cuboid corresponding to a particular alignment can be represented as a series of coordinate triplets (x, y, z). The path depicted in Figure 5.3 corresponds to a global alignment, as it starts at position $(0, 0, 0)$ and ends at position (m, n, o), where m, n, and o correspond to the lengths of the aligned sequences: $(0, 0, 0) \rightarrow (1, 1, 1) \rightarrow (2, 2, 1) \rightarrow (3, 2, 1) \rightarrow (4, 3, 2)$. Notice that for each dimension the current coordinate indicates how many residues of the relevant sequence have been incorporated into the alignment up to that point.

Now consider the alignment of globin sequences shown in Figure 5.1. If we number sequences from top to bottom $S_1, .., S_7$, and write the hyperlattice coordinates in the same order, the path implied by the multiple sequence alignment is: $(0, 0, 0, 0, 0, 0, 0) \rightarrow (0, 0, 0, 0, 1, 0, 0) \rightarrow \ldots \rightarrow (0, 0, 0, 0, 8, 0, 0) \rightarrow (1, 1, 0, 0, 9, 0, 1) \rightarrow (2, 2, 1, 1, 10, 1, 2) \rightarrow \ldots \rightarrow (146, 146, 141, 141, 149, 153, 153)$. The number of vertexes in the hyperlattice is $\prod_{i=1}^{n}(|S_i| + 1)$. This number increases rapidly with n and for our example alignment it is of the order of 10^{15}.

While the volume of our hyperlattice increases by a factor proportional to the length of each additional sequence, the time spent at each vertex also increases. This is because the number of neighboring vertexes that can influence the score at the current vertex increases, as illustrated in Figure 5.4. For two sequences we have already seen that the current score is formed as a function of three neighboring scores.

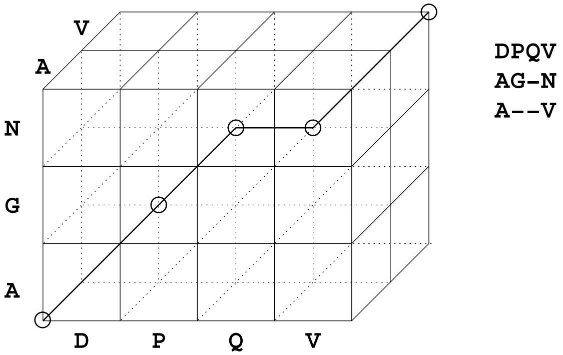
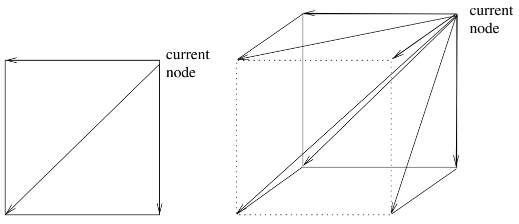


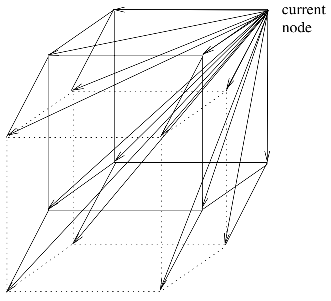
Fig. 5.3. Cuboid for the alignment of three sequences. The three sequences are written along the three axes of the hyperlattice. The path corresponding to the alignment shown on the right visits the circled vertexes.

For three sequences seven scores need to be considered, and for four sequences 15 scores. In general, each new matrix entry is a function of $2^n - 1$ scores, where n is the number of sequences to be aligned. So the total time needed to align n sequences by “naïve” dynamic programming is $O(2^n \prod_{i=1}^{n} |S_i|)$.



2 sequences: 3 vertices to visit

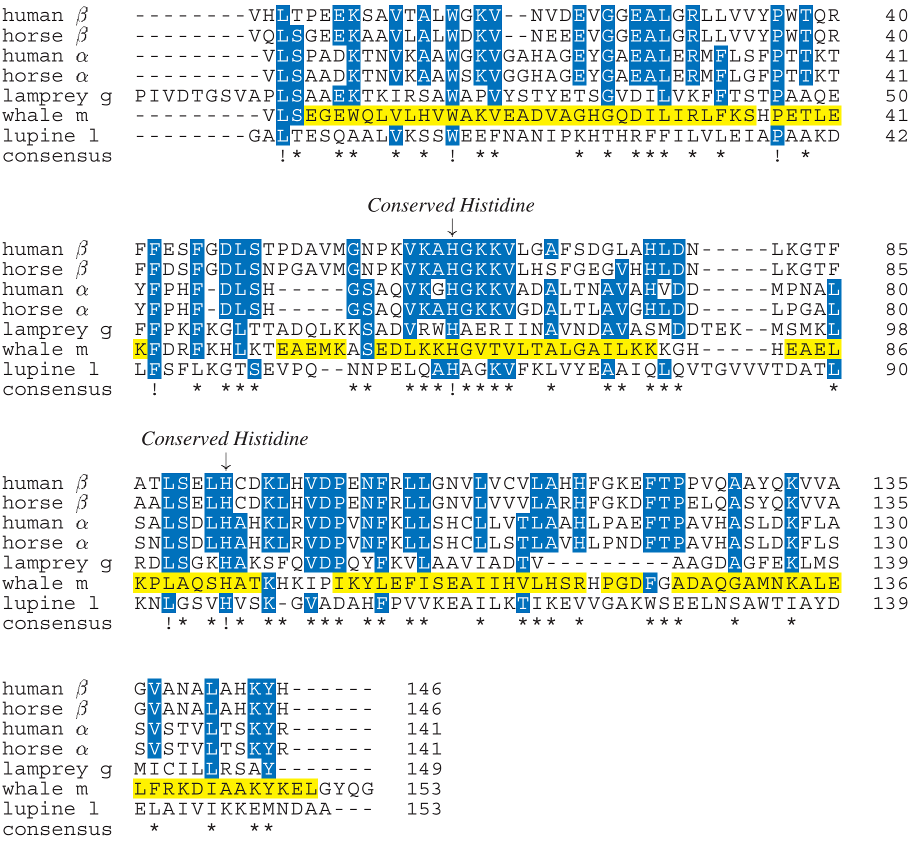
3 sequences: 7 vertices to visit



4 sequences: 15 vertices to visit

Fig. 5.4. The number of vertexes that need to be visited by a “naïve” optimal alignment algorithm at each vertex in the hyperlattice as a function of the number of sequences to be aligned.

We called the approach to multiple alignment as presented so far "naïve" because the algorithm can be drastically accelerated by considering only those vertexes in the hyperlattice that potentially lie on the path of an optimal alignment [35]. If the sequences align well, this will be a narrow "tube" along the main diagonal of the hyperlattice. The program `msa` implements such an improved optimal alignment algorithm [169, 98]. Running the program on our data set of seven globins on a 1 GH pentium 4 computer with 250 MB RAM took 0.2 seconds. The resulting multiple alignment is shown in Figure 5.5 and is very similar to our original example obtained using a heuristic method (Fig. 5.1). However, there are four differences between the two alignments, which all occur around the three largest gaps in the alignments.



```
human β     --------VHLTPEEKSAVTALWGKV--NVDEVGGEALGRLLVVYPWTQR    40
horse β     --------VQLSGEEKAAVLALWDKV--NEEEVGGEALGRLLVVYPWTQR    40
human α     ---------VLSPADKTNVKAAWGKVGAHAGEYGAEALERMFLSFPTTKT    41
horse α     ---------VLSAADKTNVKAAWSKVGGHAGEYGAEALERMFLGFPTTKT    41
lamprey g   PIVDTGSVAPLSAAEKTKIRSAWAPVYSTYETSGVDILVKFFTSTPAAQE    50
whale m     ---------VLSEGEWQLVLHVWAKVEADVAGHGQDILIRLFKSHPETLE    41
lupine l    --------GALTESQAALVKSSWEEFNANIPKHTHRFFILVLEIAPAAKD    42
consensus             !*  **  * * ! **     * * *** * *   ! *

                         Conserved Histidine
                                  ↓
human β     FFESFGDLSTPDAVMGNPKVKAHGKKVLGAFSDGLAHLDN-----LKGTF    85
horse β     FFDSFGDLSNPGAVMGNPKVKAHGKKVLHSFGEGVHHLDN-----LKGTF    85
human α     YFPHF-DLSH-----GSAQVKGHGKKVADALTNAVAHVDD-----MPNAL    80
horse α     YFPHF-DLSH-----GSAQVKAHGKKVGDALTLAVGHLDD-----LPGAL    80
lamprey g   FFPKFKGLTTADQLKKSADVRWHAERIINAVNDAVASMDDTEK--MSMKL    98
whale m     KFDRFKHLKTEAEMKASEDLKKHGVTVLTALGAILKKKGH-----HEAEL    86
lupine l    LFSFLKGTSEVPQ--NNPELQAHAGKVFKLVYEAAIQLQVTGVVVTDATL    90
consensus    !  * ***      **  ***!****  *   **  ***          *

            Conserved Histidine
                  ↓
human β     ATLSELHCDKLHVDPENFRLLGNVLVCVLAHHFGKEFTPPVQAAYQKVVA   135
horse β     AALSELHCDKLHVDPENFRLLGNVLVVVLARHFGKDFTPELQASYQKVVA   135
human α     SALSDLHAHKLRVDPVNFKLLSHCLLVTLAAHLPAEFTPAVHASLDKFLA   130
horse α     SNLSDLHAHKLRVDPVNFKLLSHCLLSTLAVHLPNDFTPAVHASLDKFLS   130
lamprey g   RDLSGKHAKSFQVDPQYFKVLAAVIADTV---------AAGDAGFEKLMS   139
whale m     KPLAQSHATKHKIPIKYLEFISEAIIHVLHSRHPGDFGADAQGAMNKALE   136
lupine l    KNLGSVHVSK-GVADAHFPVVKEAILKTIKEVVGAKWSEELNSAWTIAYD   139
consensus     !* *!* ** *** ** **   *  *** *    ***   *   *

human β     GVANALAHKYH------   146
horse β     GVANALAHKYH------   146
human α     SVSTVLTSKYR------   141
horse α     SVSTVLTSKYR------   141
lamprey g   MICILLRSAY-------   149
whale m     LFRKDIAAKYKELGYQG   153
lupine l    ELAIVIKKEMNDAA---   153
consensus    *   *  **
```

Fig. 5.5. Same as Figure 5.1, except that the latter was computed using the heuristic program `clustalw` [246], while the optimal program `msa` [98] was used to obtain the present alignment.

In order to determine the minimum score a vertex might have and still be on the path corresponding to an optimal alignment, `msa` first uses a heuristic alignment algorithm before beginning the search for the optimal alignment [169, 98]. Hence heuristic alignment methods appear to be a prerequisite for practical optimal multiple sequence alignment algorithms. Further, they are by far the most widely used class of multiple alignment methods and we turn to them next.

5.3 Heuristic Multiple Alignment

Heuristic multiple sequence alignment programs proceed in essence by computing only pairwise alignments. The trick is to compute these pairwise alignments in an order corresponding to the evolutionary relationships of the sequences to be aligned. Perhaps the most popular approach to heuristic alignment is known as progressive alignment. The method was developed in 1987 by Feng and Doolittle [74] and is the basis of such programs as `clustalw`, which we used to compute the alignment shown in Figure 5.1.

At the time when Feng and Doolittle published their proposal, the standard method for constructing multiple sequence alignments was to generate all pairwise alignments and then to assemble the multiple alignment by hand without considering the evolutionary relationships of the sequences. The authors commented on this: "It seems folly to us that a gap should be discarded in an alignment of two closely related sequences merely because an alignment with some distantly related sequence might be improved" [74, p. 352]. Since similar sequences can be aligned more reliably than dissimilar sequences, the alignment of similar sequences should be given more weight in the overall alignment scheme. This increased weight is achieved by aligning the sequences progressively, starting from the most similar pair and following the rule "once a gap, always a gap" [74]. In other words, gaps introduced early in the alignment process are not changed later on.

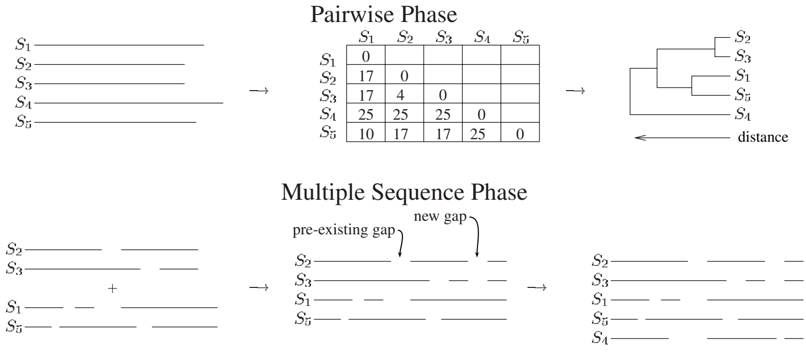


Fig. 5.6. Progressive multiple sequence alignment.

Figure 5.6 gives a graphical summary of the progressive alignment method. First, all pairwise alignments are constructed and the pairwise distances between the sequences are calculated from these alignments. A simple distance measure would be the number of pairwise mismatches. The order in which individual sequences or sets of aligned sequences progressively join the target alignment is then determined by the construction of a cluster diagram from the distance matrix. In this cluster diagram, also referred to as a “guide tree” [246], leaves represent sequences and internal nodes represent alignments. Starting from the most similar pair of sequences, the algorithm works from the leaves to the root of the guide tree. In the example shown in Figure 5.6 S_2 and S_3 are the two most similar sequences. They get aligned first, followed by S_1 and S_5. The next node up the tree clusters the two pairwise alignments formed so far and in the final step S_4 is added.

With the aid of a scoring scheme and a pairwise alignment method it is not difficult to align pairs of alignments and an example of this is shown in Figure 5.7. However, during the multiple sequence phase of heuristic alignment, the intermediate alignments are often represented as so called *profiles* (Chapter 6) rather than the standard alignments suggested by Figure 5.6.

Alignments to be aligned:

1. `AACGT`
   `A-CGT`
2. `AAGT`
   `A-GT`

Result:

`AACGT`
`A-CGT`
`AA-GT`
`A--GT`

	- -	A A	A -	C C	G G	T T
- -	**0**	← −3	← −6	← −9	← −12	← −15
A A	↑ −3	↖ **6**	← 3	← 0	← −3	← −6
A -	↑ −6	↑ 3	↖ **3**	← **0**	← −3	← −6
G G	↑ −9	↑ 0	↑ 0	↖ 1	↖ **6**	← 3
T T	↑ −12	↑ −3	↑ −3	↑↖ −2	↑ 3	↖ **12**

Fig. 5.7. Global alignment of two alignments using dynamic programming. Gaps in the existing alignments are ignored, that is, their alignment carries a score of zero. In this example match = 1, mismatch = −1, and gap = −1. The traceback path corresponding to the final alignment is marked in bold.

5.4 Summary

Multiple sequence alignments are often computed for known members of a protein family. The aim is to discover sequence motifs that are conserved across the members of the protein family in order to infer the functionally important protein domains. Like pairwise sequence alignments, multiple sequence alignments can be computed using optimal or heuristic methods. In practice most multiple sequence alignments are computed using heuristic methods. These proceed by reducing the multiple alignment problem to a series of pairwise alignments. The order in which these pairwise alignments are fused into the multiple alignment is crucial for the success of the algorithm. The pairwise alignments of the input sequences are used to calculate pairwise distances. This in turn serves as input data for the construction of a guide tree, which determines the order in which sequences are added to the multiple alignment.

5.5 Further Reading

A number of comprehensive surveys of multiple alignment methods have been published over the years [176, 247, 62]. In addition, Chapter 14 in Gusfield's textbook treats many of the algorithmic aspects of multiple sequence alignment [99].

5.6 Exercises and Study Questions

5.1. How would you interpret the fact that at position 11 of the globin alignments in Figures 5.1 and 5.5 all seven sequences have a leucine (`L`)?

5.2. Try to verify that the myoglobin molecule depicted in Figure 5.2 contains the eight alpha helices marked in the myoglobin sequence of the globin multiple sequence alignment (Fig. 5.1).

5.3. Write down the hyperlattice for aligning the three sequences `AATG`, `ATG`, and `TG` and trace the path corresponding to the alignment

```
AATG
A-TG
--TG
```

5.4. Write the alignment path traced in the previous exercise as a series of triplets of coordinates in the hyperlattice.

5.5. Calculate the score of the alignment using the sum-of-pairs method and mismatch $= -1$, match $= 1$, gap $= -1$. Remember that the alignment of two gaps is ignored (scored as zero).

5.6. Can you think of an alternative cooptimal path (and hence alignment)?

5.7. In the hyperlattice corresponding to the alignment in Figure 5.1, what are the coordinates of the vertex following position $(1, 1, 0, 0, 9, 0, 1)$?

5.8. What is the exact number of vertexes in the hyperlattice necessary for computing an alignment of the seven globins shown in Figure 5.1 by dynamic programming?

5.9. Given that each operation in a "naïve" version of the optimal multiple alignment algorithm takes one nanosecond, how many sequences of length 100 can be aligned in a decade?

5.10. Find all the differences between the alignments shown in Figures 5.1 and 5.5.

5.11. The first step in "progressive" alignment algorithms is the formation of all possible pairwise alignments. Given n sequences to be aligned, how many such pairwise alignments need to be computed?

5.12. Aligning the translation of a set of DNA sequences using `clustalw` is faster than aligning the corresponding DNA sequences. Why is this the case and how much faster is protein alignment compared to DNA alignment?

5.13. Use the dynamic programming matrix shown in Figure 5.7 to determine an alignment that is cooptimal to the one shown.

6

Sequence Profiles and Hidden Markov Models

> *Biologists should realize that before long we shall have a subject which might be called 'protein taxonomy'—the study of the amino acid sequences of the proteins of an organism and the comparison of them between species.*
>
> *Francis H. C. Crick [42, p. 142]*

Sequence profiles summarize the information contained in a multiple sequence alignment. Such a summary can then be used to search for other members of the sample contained in the underlying multiple sequence alignment.

Hidden Markov models are stochastic models originally developed in the context of speech recognition. It was later realized that sequence profiles can be formulated as hidden Markov models, thereby introducing powerful new methods to sequence analysis. In this chapter we first introduce sequence profiles before turning our attention to their reformulation in the context of hidden Markov models.

6.1 Profile Analysis

Profile analysis is centered on protein families, which are groups of proteins that have similar functions. An example of such a protein family is the globins, all of which are involved in oxygen transport. A phylogeny of seven members of this protein family is depicted in Figure 6.1. Members of a protein family may or may not all be connected by significant pairwise alignments. For example, if we carry out a standard BLAST search with human α-globin as the query sequence marked in Figure 6.1, we get significant hits up to the evolutionary distance of sperm whale myoglobin (boxed sequences in Figure 6.1). However, we observe no hit to leghemoglobin. Hence, in such a pairwise search, the homology between α-globin and leghemoglobin would go undetected.

Database search programs such as `blastp`, a member of the BLAST package for comparing pairs of protein sequences, employ score matrices such as the PAM and BLOSUM series. These matrices are position independent, that is, they are applied with equal weight to every position in an alignment. In contrast, profiles are position-specific score matrices that capture the primary structure of a set of proteins belonging to the same family. In order to demonstrate the power of profile analysis, consider again our globin example. PROSITE is a database containing profiles characteristic of protein families [226]. A scan of human α-globin against PROSITE

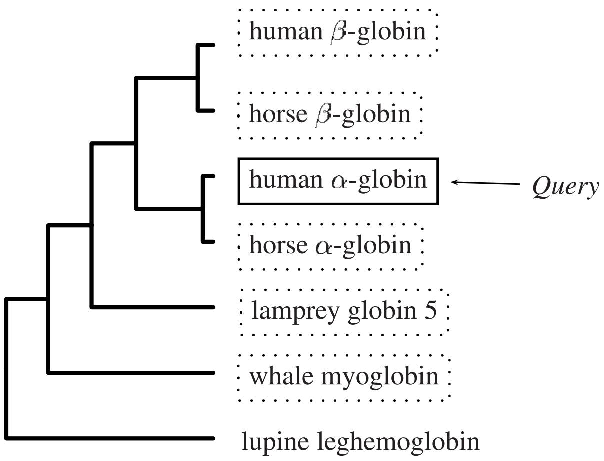


Fig. 6.1. Phylogeny of seven members of the globin protein family. Boxed sequences are those detected by comparing human α-globin against the complete set of seven sequences.

returns a match to the globin profile, and a match to the plant-globin profile, which contains leghemoglobin from yellow lupine (Fig. 6.1). Hence, profile analysis can detect homologies across greater evolutionary distances than standard pairwise alignment.

Profile analysis is carried out in two steps: (i) construction of the profile and (ii) comparison of the profile with a single sequence or a database of sequences. Profile construction always starts from an alignment of multiple homologous sequences. The simplest method for extracting a profile from such a multiple sequence alignment is to count at each position in the alignment the frequency of each of the possible 20 amino acids. The result of subjecting the first 20 positions of our globin alignment (Fig. 5.1) to such an analysis is shown in Figure 6.2. The profile consists of 20 columns, one for each possible amino acid, and a row for each position in the alignment. For example, at position 10 we find three valines (`V`) and one each of alanine (`A`), histidine (`H`), proline (`P`), and glutamine (`Q`).

More elaborate scoring schemes that incorporate the standard amino acid score matrices have been proposed [90]. For example, the weighted score $M(p, a)$ of amino acid a at position p can be defined as

$$M(p, a) = \sum_{b=1}^{20} w(p, b)s(a, b),$$

where $w(p, b)$ is the weight of amino acid b at position p and $s(a, b)$ is the score of aligning amino acids a and b as determined from an amino acid substitution matrix. If the weighting scheme is linear, $w(p, b)$ is the frequency of amino acid b in column p of the multiple alignment from which the profile is computed. With a logarithmic weighting scheme the log of the frequency would be used with the added proviso that

Position	A	C	D	E	F	G	H	I	K	L	M	N	P	Q	R	S	T	V	W	Y
1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
4	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
6	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
9	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	2	0	0
10	$\boxed{1}$	0	0	0	0	0	$\boxed{1}$	0	0	0	0	0	$\boxed{1}$	$\boxed{1}$	0	0	0	$\boxed{3}$	0	0
11	0	0	0	0	0	0	0	0	0	7	0	0	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5	2	0	0	0
13	2	0	0	2	0	1	0	0	0	0	0	0	2	0	0	0	0	0	0	0
14	3	0	0	2	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0
15	0	0	2	4	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
16	1	0	0	0	0	0	0	0	5	0	0	0	0	0	0	0	0	0	1	0
17	2	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	3	0	0	0
18	2	0	0	0	0	0	0	0	1	2	0	2	0	0	0	0	0	0	0	0
19	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	6	0	0
20	0	0	0	0	0	0	0	0	3	2	0	0	0	0	1	0	1	0	0	0

Fig. 6.2. Frequency count of the first 20 positions of the globin alignment shown in Figure 5.1. The boxed entries are discussed in the text.

if an amino acid is missing from a particular column, it is counted as being present once. The result of applying the linear scoring scheme together with an amino acid score matrix (Table 6.1) to the first 20 positions of our globin alignment is shown in Figure 6.3. The boxed entries are the same as those boxed in the simple profile of Figure 6.2. In order to understand the entries in the matrix shown in Figure 6.3, consider its first line. Since there is only proline (`P`) present in the first column of the sequence alignment, all entries in the profile are simply $s(a_i, \texttt{P})/n$, where n is the number of sequences, i.e. seven in our case. For example, $s(\texttt{A}, \texttt{P}) = 5$ and hence the entry in the first line and the first column of the profile is $5/7 \approx 0.7$.

Notice that the last column of the profile gives a position-specific gap-score. The first nine positions of the alignment contain gaps (Fig. 5.1), which is reflected by a jump in the gap scores between positions < 10 and those ≥ 10 (Fig. 6.3).

Aligning a sequence to a profile can be done using the standard local dynamic programming technique. The only difference is that scores are now looked up in the profile. Consider for example the multiple DNA sequence alignment in Figure 6.4A. Given a score scheme of match $= 1$ and mismatch $= -1$, we obtain the corresponding profile shown in Figure 6.4B. Figure 6.5 shows how this profile can be aligned with the example sequence `ACCT`. The sequence is written along the horizontal axis of the dynamic programming matrix and the profile along the vertical axis. We carry out a local alignment, so the first row and column are initialized with zeros. As before, gap extensions are scored as -1, and gap opening as 0. The score of a match

Table 6.1. Amino acid substitution matrix for profile analysis [89]. Match scores are shown in bold.

	A	B	C	D	E	F	G	H	I	K	L	M	N	P	Q	R	S	T	V	W	Y	Z
A	**15**	2	3	3	3	-5	6	-1	0	0	-1	0	2	5	2	-3	4	4	2	-8	-3	2
B	2	**11**	-4	11	6	-6	6	4	-2	4	-5	-3	11	1	5	1	3	2	-2	-6	-3	6
C	3	-4	**15**	-5	-6	-1	2	-1	2	-6	-8	-6	-3	1	-6	-3	6	2	2	-12	10	-6
D	3	11	-5	**15**	10	-10	6	4	-2	3	-5	-4	6	1	6	0	2	2	-2	-11	-5	8
E	3	6	-6	10	**15**	-6	5	4	-2	3	-3	-2	5	1	6	0	2	2	-2	-11	-5	11
F	-5	-6	-1	-10	-6	**15**	-6	-1	6	-6	12	5	-5	-6	-8	-5	-3	-3	2	12	13	-6
G	6	6	2	6	5	-6	**15**	-2	-3	-1	-5	-3	4	3	2	-3	6	4	2	-10	-6	3
H	-1	4	-1	4	4	-1	-2	**15**	-3	1	-2	-3	5	2	6	5	-2	-1	-3	-1	3	5
I	0	-2	2	-2	-2	6	-3	-3	**15**	-2	8	6	-3	-2	-3	-3	-1	2	11	-5	1	-2
K	0	4	-6	3	3	-6	-1	1	-2	**15**	-3	2	4	1	4	8	2	2	-2	1	-6	4
L	-1	-5	-8	-5	-3	12	-5	-2	8	-3	**15**	12	-4	-3	-1	-4	-4	-1	8	5	3	-2
M	0	-3	-6	-4	-2	5	-3	-3	6	2	12	**15**	-3	-2	0	2	-3	0	6	-3	-1	-1
N	2	11	-3	6	5	-5	4	5	-3	4	-4	-3	**15**	0	4	1	3	2	-3	-3	-1	4
P	5	1	1	1	1	-6	3	2	-2	1	-3	-2	0	**15**	3	3	4	3	1	-8	-8	2
Q	2	5	-6	6	6	-8	2	6	-3	4	-1	0	4	3	**15**	4	-1	-1	-2	-5	-6	11
R	-3	1	-3	0	0	-5	-3	5	-3	8	-4	2	1	3	4	**15**	1	-1	-3	13	-6	2
S	4	3	6	2	2	-3	6	-2	-1	2	-4	-3	3	4	-1	1	**15**	3	-1	3	-4	0
T	4	2	2	2	2	-3	4	-1	2	2	-1	0	2	3	-1	-1	3	**15**	2	-6	-3	1
V	2	-2	2	-2	-2	2	2	-3	11	-2	8	6	-3	1	-2	-3	-1	2	**15**	-8	-1	-2
W	-8	-6	-12	-11	-11	12	-10	-1	-5	1	5	-3	-3	-8	-5	13	3	-6	-8	**15**	11	-8
Y	-3	-3	10	-5	-5	13	-6	3	1	-6	3	-1	-1	-8	-6	-6	-4	-3	-1	11	**15**	-6
Z	2	6	-16	8	11	-6	3	5	-2	4	-2	-1	4	2	11	2	0	1	-2	-8	-6	**11**

Position	A	C	D	E	F	G	H	I	K	L	M	N	P	Q	R	S	T	V	W	Y	-
1	0.7	0.1	0.1	0.1	-0.8	0.4	0.2	-0.2	0.1	-0.4	-0.2	0.0	2.1	0.4	0.4	0.5	0.4	0.1	-1.1	-1.1	1.2
2	0.0	0.2	-0.2	-0.2	0.8	-0.4	-0.4	2.1	-0.2	1.1	0.8	-0.4	-0.2	-0.4	-0.4	-0.1	0.2	1.5	-0.7	0.1	1.2
3	0.2	0.2	-0.2	-0.2	0.2	0.2	-0.4	1.5	-0.2	1.1	0.8	-0.4	0.1	-0.2	-0.4	-0.1	0.2	2.1	-1.1	-0.1	1.2
4	0.4	-0.7	2.1	1.4	-1.4	0.8	0.5	-0.2	0.4	-0.7	-0.5	0.8	0.1	0.8	0.0	0.2	0.2	-0.2	-1.5	-0.7	1.2
5	0.5	0.2	0.2	0.2	-0.4	0.5	-0.1	0.2	0.2	-0.1	0.0	0.2	0.4	-0.1	-0.1	0.4	2.1	0.2	-0.8	-0.4	1.2
6	0.8	0.2	0.8	0.7	-0.8	2.1	-0.2	-0.4	-0.1	-0.7	-0.4	0.5	0.4	0.2	-0.4	0.8	0.5	0.2	-1.4	-0.8	1.2
7	0.5	0.8	0.2	0.2	-0.4	0.8	-0.2	-0.1	0.2	-0.5	-0.4	0.4	0.5	-0.1	0.1	2.1	0.4	-0.1	0.4	-0.5	1.2
8	0.2	0.2	-0.2	-0.2	0.2	0.2	-0.4	1.5	-0.2	1.1	0.8	-0.4	0.1	-0.2	-0.4	-0.1	0.2	2.1	-1.1	-0.1	1.2
9	3.5	1.2	0.7	0.5	-1.0	3.5	-1.2	2.7	-0.7	1.4	1.2	0.0	1.4	0.0	-1.7	1.1	1.7	4.8	-4.8	-1.5	1.2
10	$\boxed{3.8}$	0.4	1.1	1.1	-2.0	2.1	$\boxed{1.8}$	3.5	0.0	2.4	1.8	0.2	$\boxed{4.0}$	$\boxed{2.8}$	0.0	0.2	1.5	$\boxed{6.1}$	-6.5	-2.4	4.5
11	-1.0	-8.0	-5.0	-3.0	12.0	-5.0	-2.0	8.0	-3.0	15.0	12.0	-4.0	-3.0	-1.0	-4.0	-4.0	-1.0	8.0	5.0	3.0	4.5
12	4.0	4.8	2.0	2.0	-3.0	5.4	-1.7	-0.1	2.0	-3.1	-2.1	2.7	3.7	-1.0	0.4	11.5	6.4	-0.1	0.4	-3.7	4.5
13	7.4	-0.2	4.8	6.1	-5.7	6.1	1.1	-1.5	1.0	-2.7	-1.5	2.5	6.4	3.4	-0.4	3.7	3.1	0.5	-9.1	-5.4	4.5
14	8.7	0.7	5.2	6.5	-5.1	7.0	0.1	-1.1	1.0	-2.5	-1.4	3.2	3.4	2.7	-1.5	5.2	3.2	0.4	-7.5	-4.1	4.5
15	2.8	-5.7	10.8	12.2	-7.4	4.8	4.2	-2.1	3.1	-3.2	-2.2	5.1	1.2	7.2	0.5	1.5	1.5	-2.0	-10.1	-5.1	4.5
16	1.0	-5.5	1.0	1.0	-3.2	-1.2	0.4	-2.1	10.8	-1.5	1.0	2.7	0.2	2.4	7.1	2.4	1.1	-2.2	1.7	-3.1	4.5
17	6.8	1.7	2.8	2.8	-4.2	4.5	-0.1	0.2	1.7	-1.4	-0.4	2.4	3.7	2.1	-0.5	4.4	7.8	1.0	-5.1	-3.5	4.5
18	4.5	-3.1	1.5	1.8	-0.2	1.2	0.7	1.1	2.4	2.4	2.8	4.2	0.7	2.0	-0.5	1.1	1.7	1.7	-1.5	-1.1	4.5
19	1.7	2.0	-2.0	-2.0	2.5	1.2	-3.0	11.5	-2.0	8.0	6.0	-3.0	0.5	-2.1	-3.0	-1.0	2.0	14.4	-7.5	-0.7	4.5
20	-0.1	-5.0	0.1	0.7	-0.2	-1.7	0.4	1.2	7.0	2.2	4.5	1.0	0.4	1.8	4.2	0.2	2.5	1.2	2.8	-3.0	4.5

Fig. 6.3. Profile of the first 20 positions of the globin alignment shown in Figure 5.1. For details see text. The profile was calculated using the EMBOSS software package [209].

between position i in the profile and j in the sequence is looked up as entry i, j in the profile. The two cooptimal best local alignments are $\binom{\texttt{AC}}{\texttt{AC}}$ and $\binom{\texttt{CT}}{\texttt{CT}}$; both carry a score of 0.8.

A

1	2	3	4	5
A	C	G	C	A
A	C	G	C	T
A	C	G	C	T
A	G	G	G	T
C	G	G	T	T

B

Position	A	C	G	T
1	0.6	-0.6	-1	-1
2	-1	0.2	-0.2	-1
3	-1	-1	1	-1
4	-1	0.2	-0.6	-0.6
5	-0.6	-1	-1	0.6

Fig. 6.4. A: Alignment of five DNA sequences with alignment positions given in the top row. **B**: Profile computed from the alignment using a score scheme of match $= 1$ and mismatch $= -1$.

Position	A	C	G	T	-	A	C	C	T
0	0	0	0	0	0	0	0	0	0
1	0.6	-0.6	-1	-1	0	↖0.6	0	0	0
2	-1	0.2	-0.2	-1	0	0	↖0.8	0	0
3	-1	-1	1	-1	0	0	0	0	0
4	-1	0.2	-0.6	-0.6	0	0	0	↖0.2	0
5	-0.6	-1	-1	0.6	0	0	0	0	↖0.8

Fig. 6.5. Local alignment between a profile (vertical axis) and a DNA sequence (horizontal axis).

We now turn to a more realistic application of profile analysis. Imagine that we have identified the six boxed globin sequences shown in Figure 6.1 by pairwise homology search. In order to look for further members of our protein family, we compute a profile from these six sequences. An alignment of leghemoglobin to this profile happens to result in a score of 725.36. Is this significant? One method to determine this is to compare the original score to the distribution of scores obtained by aligning unrelated proteins to the profile. Alternatively, the sequence of leghemoglobin can be repeatedly randomized and realigned to the profile. The result of 1,000 such randomizations is shown in Figure 6.6A, which implies a marginal significance level of 0.009. If we now iterate the analysis and include leghemoglobin in the profile computation, we get the result shown in Figure 6.6B. It is clear that in this case the original score is due to highly significant homology between leghemoglobin and the globin profile. The superior sensitivity of profiles relative to pairwise comparisons is the reason for using profiles as intermediate steps in heuristic multiple sequence alignment algorithms such as `clustalw` [246].

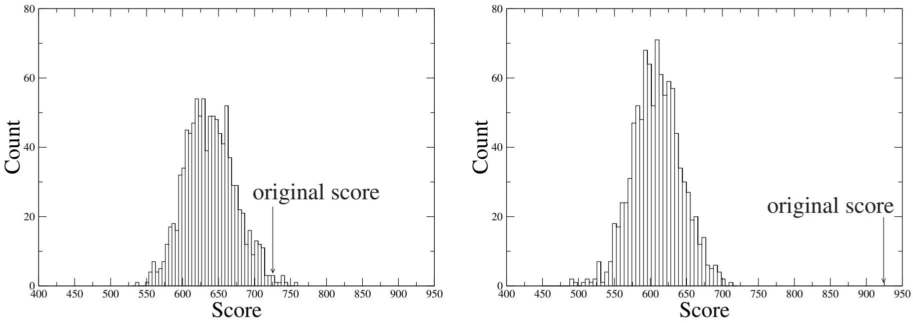


Fig. 6.6. The sequence of leghemoglobin was shuffled 1,000 times and aligned to the globin profile. The resulting distribution of alignment scores can then be compare to the score of the alignment between the profile and the unshuffled leghemoglobin sequence (original score). **A**: Profile computed from the six boxed globin sequences shown in Figure 6.1. **B**: Profile computed from all seven globins shown in Figure 6.1. Profile generation and profile alignment was carried out using the EMBOSS programs `prophecy` and `prophet`, respectively [209].

We have already mentioned the limited sensitivity of classical pairwise alignment programs such as `blastp`. However, the BLAST package also contains a program based on profile searches called `psi-blast` [10]. It works by iterating pairwise searches. After each round of pairwise searching, a profile is calculated from the set of homologous sequences and used as query for searching in the next round. The iteration stops once no new homologues are found. When using human α-globin as query against the SwissProt database, `psi-blast` converges after five rounds of iteration with a results list that contains a highly significant hit to leghemoglobin. No such hit was contained in the results list after the first round of searching. Hence `psi-blast` considerably extends the reach of database searches in homology space.

Profile analysis had been applied to sequence analysis problems for a few years when it was realized that profiles can be rewritten as hidden Markov models [155]. These are stochastic models that can be applied to the analysis of diverse kinds of sequential data, most notably human speech and biological sequences.

6.2 Hidden Markov Models

Hidden Markov models (HMMs) are a class of stochastic models that are widely used in computational biology. Perhaps their most successful applications are to gene prediction and homology detection. Homology detection by HMMs is covered in Section 6.3, while gene prediction is the topic of Chapter 7.

In the following we first explain what a hidden Markov model is and how it can be used to analyze sequential data including DNA and protein sequences. This is followed by a survey of a special kind of HMM, profile HMM, which is derived from a multiple sequence alignment. Profile HMMs are widely applied in protein classification and can themselves be used to rapidly calculate multiple sequence alignments.

Hidden Markov models were made accessible to the scientific community in a classical tutorial by Rabiner [205] and we follow his treatment. Consider a system, such as your body, which can take one of $N = 2$ distinct states, e.g. "healthy" and "ill". Let a certain state at time t be called q_t. A discrete, n-th order Markov chain has the property that a state at time t depends on the system's states at times $t-1, t-2, \ldots, t-n$. Here we will only be concerned with first order Markov chains. Dealing with a first order Markov chain means that the probabilistic description of a state sequence is truncated to the present and the previous state:

$$P(q_t = S_j \mid q_{t-1} = S_i, q_{t-2} = S_k, \ldots) = P(q_t = S_j \mid q_{t-1} = S_i).$$

As the system evolves from time t to time $t+1$, it switches between states according to time-independent probabilities of the form

$$a_{ij} = P(q_t = S_j \mid q_{t-1} = S_i),$$

where S_i and S_j may be identical. Figure 6.7 depicts our healthy *vs.* ill example as a Markov chain with two states. The transition probabilities marked in Figure 6.7 might alternatively be written as

$$A = \{a_{ij}\} = \begin{pmatrix} 0.99 & 0.01 \\ 0.3 & 0.7 \end{pmatrix}.$$

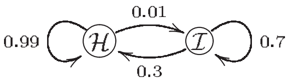


Fig. 6.7. Markov chain with two states "healthy" ($\mathcal{H}$) and "ill" ($\mathcal{I}$).

Given that you are healthy now, we can ask, what is the probability that you will be healthy – healthy – healthy – ill – ill over the next five days? The answer is $0.99^3 \cdot 0.01 \cdot 0.7 \approx 0.007$. Perhaps more interestingly, we can ask, what is the average duration, $\bar{d}_i$, of state i? This can be found by summing the probabilities of all possible durations of a certain state

$$\bar{d}_i = \sum_{d=1}^{\infty} d \cdot a_{ii}^{d-1}(1 - a_{ii}) = \frac{1}{1 - a_{ii}}.$$

That is, our Markov chain in Figure 6.7 implies an expected 100 consecutive days of health and $10/3$ consecutive days of illness, if states can switch once a day.

In many situations it is necessary to distinguish between a system's state and the value of an observable quantity which depends on that state. For example, a common indicator of health or otherwise is our body temperature. Hence, we call "healthy"

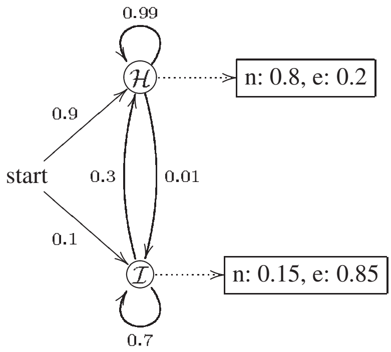


Fig. 6.8. Hidden Markov model with hidden states "healthy" ($\mathcal{H}$) and "ill" ($\mathcal{I}$). Each hidden state emits one of two possible observable states, normal (n) and elevated (e) body temperature with the indicated probabilities.

and "ill" *hidden* states which emit the observable states "normal" (n) or "elevated" (e) body temperature (Fig. 6.8). Such a model can be used to generate a series of binary temperature recordings: nnneennnnnn... Before we learn how to generate such an observation sequence using our HMM, let us summarize its elements:

1. N: the number of states $S = \{S_1, S_2, \ldots, S_N\}$; a state at time t is referred to as q_t
2. M: the number of observational symbols, that is, the size of alphabet $V = \{v_1, v_2, \ldots, v_M\}$
3. $A = \{a_{ij}\}$: state transition probabilities $a_{ij} = P(q_t = S_j \mid q_{t-1} = S_i)$, $1 \leq i, j, \leq N$
4. $B = \{b_j(k)\}$: observation symbol probabilities; $b_j(k) = P(v_k \text{ at } t \mid q_t = S_j)$, $1 \leq j \leq N, 1 \leq k \leq M$
5. $\pi = \{\pi_i\}$: initial state probabilities; $\pi = P(q_1 = S_i)$, $1 \leq i \leq N$

The complete parameter set is conveniently referred to as

$$\lambda = (A, B, \pi).$$

If we now return to our task of generating an observation sequence

$$O = O_1, O_2, \ldots O_T,$$

where all $O_t \in V$, we can use the following procedure:

1. choose q_1 according to π; $t \leftarrow 1$
2. choose $O_t = v_k$ according to $b_i(k)$
3. change to new state $q_{t+1} = S_j$ according to a_{ij}; $t \leftarrow t + 1$
4. if $t < T$, goto 2; else stop

We demonstrate the application of this procedure using the HMM of Figure 6.8 and the following list of random numbers:

r_1	r_2	r_3	r_4	...
0.66287	0.505421	0.495442	0.568873	...

The algorithm starts at "start" in Figure 6.8. By sequentially using up the random numbers and following the steps in the algorithm it yields

Step 1: $r_1 < 0.9$, therefore $q_1 = \mathcal{H}$
Step 2: $r_2 < 0.8$, therefore $O_1 = \text{n}$
Step 3: $r_3 < 0.99$, therefore $q_2 = \mathcal{H}$
Step 2: $r_4 < 0.8$, therefore $O_2 = \text{n}$
...

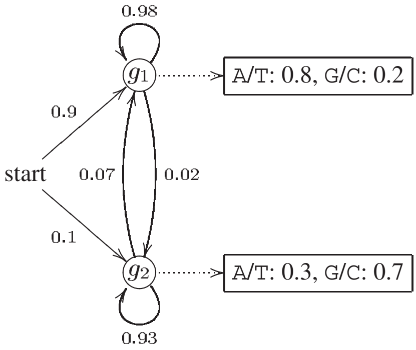
Now that we are acquainted with the nuts and bolts of HMMs, we can start using them to solve interesting problems. To fix our ideas, consider the HMM in Figure 6.9, which models in a simplified fashion G/C- and A/T-rich genomic regions. Figure 6.9 also contains an observation sequence, a DNA sequence. It is convenient to think of such a sequence as having been generated by a HMM, even though in most situations of practical relevance it would be real sequence data. The metaphor of a HMM generating a given biosequence is a shorthand for saying that the HMM is believed to model important properties of this sequence. Given such a model and the corresponding empirical data we can solve the following three fundamental problems for HMMs:

1. The scoring problem: given O and λ, compute $P(O \mid \lambda)$; in the context of our example in Figure 6.9 we wish to know the probability that a given DNA sequence has been generated by a known HMM.
2. The detection problem: given O and λ, find the optimal state sequence; for our example this means that we wish to know which parts of the sequence are A/T-rich and which are G/C-rich.
3. The training problem: given O and an initial λ, maximize $P(O \mid \lambda)$ by adjusting the model's transition and emission probabilities (not its architecture of hidden states); for our example we would start with DNA sequence data containing A/T- and G/C-rich regions and would try to determine the corresponding emission and transition probabilities.

All three problems can be solved efficiently and in the following we explain the solutions to the scoring and the detection problem. Readers interested in the slightly more advanced training problem are referred to Rabiner's article [205].

A naïve method for solving the scoring problem would be to write down all possible state sequences, calculate the probability of each one and choose the state sequence with the highest probability. However, the number of state sequences is N^T, which renders this approach unfeasible. Fortunately, a bottom up approach similar to that encountered in the context of computing the number of possible global alignments (Section 2.5) can also be applied in this case and is known as *forward procedure*. We fill a two-dimensional table with N rows and T columns and determine its entries $\alpha_t(i)$ as follows:

1. initialization: $\alpha_1(i) = \pi_i b_i(O_1)$



Observed	T	G	C	T	A	T	T	G	G	T
Hidden	g_2	g_2	g_1	g_1	g_1	g_1	g_1	g_1	g_1	g_1

Fig. 6.9. Hidden Markov model of A/T and G/C-rich genomic regions g_1 and g_2, respectively. The sequence of observed and hidden states was generated using this model.

2. recursion: $\alpha_{t+1}(j) = b_j\left(O_{t+1}\right) \sum_{i=1}^{N} \alpha_t(i) a_{ij}$, $1 \leq t \leq T-1, 1 \leq j \leq N$
3. termination: $P(O \mid \lambda) = \sum_{i=1}^{N} \alpha_T(i)$

Figure 6.10 shows the first two columns in the table of $\alpha_t(i)$ values corresponding to the data and HMM of Figure 6.9.

	T	G	...
g_1	0.72	0.14154	...
g_2	0.03	0.02961	...

Fig. 6.10. Calculation of the probability of an observation sequence using the forward procedure. The data and underlying HMM are shown in Figure 6.9.

In contrast to the scoring problem, the detection problem contains an element of optimization: we wish to find the state sequence that *best* explains the observation sequence. A dynamic programming approach known as *Viterbi algorithm* can be used to efficiently find the most likely sequence of hidden states. This algorithm is based on a similar table as employed in the forward procedure.

1. initialization: $\delta_1(i) = \pi_i b_i\left(O_1\right)$, $1 \leq i \leq N$
2. recursion: $\delta_t(j) = b_j\left(O_t\right) \max_{1 \leq i \leq N}\left(\delta_{t-1}(i) a_{ij}\right)$, $2 \leq t \leq T$, $1 \leq j \leq N$; in addition, keep back pointer to $\max_{1 \leq i \leq N}\left(\delta_{t-1}(i) a_{ij}\right)$
3. back tracking: Start from $\max_{1 \leq i \leq N}\left(\delta_T(i) a_{ij}\right)$ and follow the back pointers

Application of this algorithm to our DNA sequence example (Fig. 6.9) yields the dynamic programming matrix of Figure 6.11. Notice that the most likely state sequence differs from the known state sequence in Figure 6.9 due to the stochastic nature of the model.

	T	G	C	T ...
g_1	**0.7200**	← **0.1411**	←**0.0277**	←**0.0217** ...
g_2	0.0300	0.0195	0.0127	0.0035 ...

Fig. 6.11. Dynamic programming matrix used in the Viterbi algorithm according to the DNA sequence and HMM shown in Figure 6.9. Back pointers for the most likely state sequence are also shown.

The probabilities in the programming tables of Figures 6.10 and 6.11 rapidly become very small and lead to numerical underflow in computer programs. The corresponding algorithms are therefore always implemented on the basis of log-probabilities.

Figure 6.9 suggests how HMMs might be useful in determining G/C-rich regions, which are also often gene-rich. However, the models we have looked at so far are unrealistically simple. In addition, they have the property of being fully connected, that is, all transition probabilities are > 0. As we will see in Section 6.3, profile HMMs differ on both counts.

6.3 Profile Hidden Markov Models

As explained in Section 6.1, profiles model protein families through position-specific residue scores as well as gap penalties. Such position-specific scores can then be used to detect more subtle homologies than is possible using pairwise alignments. In the mid 1990s it was realized that profiles could be rewritten as HMMs usually known as profile HMMs, thereby making profile analysis more rigorous [155].

Profile HMMs consist of the three elements of an alignment: matches, insertions, and deletions. As shown in Figure 6.12, these elements are interpreted as hidden states in the HMM. The delete state is the simplest state; it emits only a single symbol, the gap symbol. The insertion state, i, emits amino acids drawn randomly from their background frequencies. The match state, finally, emits amino acids according to their frequencies at a given alignment position. It is important to realize that the match state is position-specific and this property distinguishes it from the position-invariant insertion and deletion states. In fact, the emissions of the match state correspond to the profile constructed from the underlying multiple sequence alignment, hence, the name profile HMMs.

Complete profile HMMs are built from a series of the elements shown in Figure 6.12 and an example model is displayed in Figure 6.13. The first thing to notice is that it is directed from a "start" to an "end" state and contains no "backward" transitions. The number of match states is equal to the length of the profile and the probabilities of the emission states are derived from the profile entries.

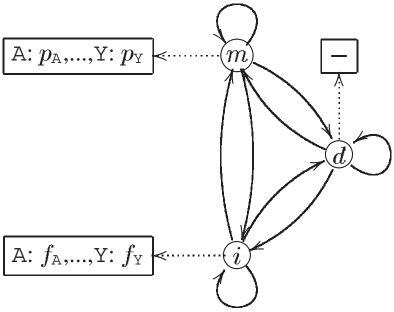


Fig. 6.12. Precursor of profile hidden Markov model. p_i: observed frequency of amino acid i; f_i: background frequency of amino acid i; d: delete state; i: insertion state; m: match state.

In contrast to profiles, which may contain arbitrary gap scores as well as pseudocounts, all free parameters of HMMs, including position-specific gap scores, are derived from the data in a consistent fashion. These parameters correspond to probabilities assigned to the arrows in Figure 6.13.

Given the complexity of protein structure, it might appear surprising that a first order rather than a higher order Markov process should be able to account for protein structure. It turns out that profile HMMs are at least as sensitive as traditional profiles in the detection of distant homologues [54]. In addition, they are not only derived from multiple sequence alignments, but can themselves be used to align hundreds of sequences very efficiently.

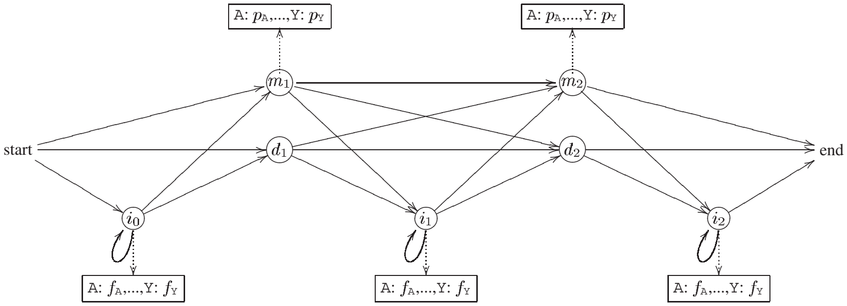
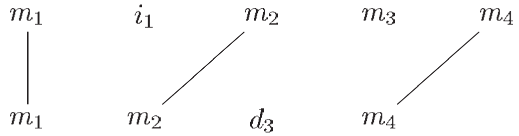


Fig. 6.13. Example profile hidden Markov model for protein sequences. Adapted from [155]. p_i: observed frequency of amino acid i; f_i: background frequency of amino acid i; d: delete state; i: insertion state; m: match state.

Searching for homologous sequences using hidden Markov models is conceptually simple: Given a set of profile HMMs characterizing protein families and an anonymous protein sequence, compare the sequence to each HMM in turn using the likelihood returned by the forward procedure as the alignment's score. A significant score then indicates which protein family the anonymous sequence belongs to.

There are databases of profile HMMs that are routinely used for this purpose. The best known of these is perhaps the protein family database PFAM [17].

While the forward procedure solves the problem of homology determination, the Viterbi algorithm can be used to compute alignments. Consider the two protein sequences $O_1 =$ `GAHYA` and $O_2 =$ `GHA` and let $S_1 = m_1, i_1, m_2, m_3, m_4$ and $S_2 = m_1, m_2, d_3, m_4$ be the corresponding most likely state sequences. The alignment implied by these state sequences is then obtained by writing amino acids emitted by the same match state in the same column. For our example the connections



yield the alignment

```
GAHYA
G-H-A
```

This procedure can be extended to an arbitrary number of sequences. For example, if we wanted to add the sequence $O_3 =$ `HYA` with the most likely state sequence $S_3 = d_1 m_2 m_3 m_4$ to our alignment, we would get

```
GAHYA
G-H-A
--HYA
```

Notice that an insertion in one sequence leads to the introduction of a gap into all other sequences. In contrast, a deletion produces a gap only in the sequence concerned.

The Viterbi algorithm runs in time proportional to the length of the profile HMM, whose number of hidden states in turn is proportional to the length of the input sequence. Hence, the alignment procedure just described runs in time proportional to the combined lengths of the input sequences. This contrasts with the standard progressive multiple sequence alignment algorithm, which runs in $O(n^2 \cdot k(k-1)/2)$, where n is the length of the k input sequences. However, in spite of this computational efficiency, HMMs are in practice rarely used for calculating multiple sequence alignments. The reason for this is that a good HMM needs to be calculated in the first place, which tends to require a large number of sequences and is computationally intensive. For most multiple sequence alignment tasks the progressive algorithm implemented in programs such as `clustalw` [246] generates more reliable results than alignments based directly on HMMs [247].

6.4 Summary

Profiles are position-specific score matrices derived from multiple sequence alignments. They are typically used for protein classification, where an unknown amino acid sequence is compared to a set of profiles characterizing known protein families. Such comparisons between profiles and anonymous sequences are often more sensitive than pairwise comparisons. Profiles can be rewritten as profile hidden Markov models. Hidden Markov models are based on Markov chains. These consist of a number of states and the probabilities of switching between them. The first order Markov chains considered in this chapter have the property that the chain's state at some point in time depends only on its direct predecessor state. In *hidden* Markov models (HMMs), the states of the Markov chain are said to be hidden and to emit observation states, for example a DNA sequence. Given such data and a HMM, there are three classical problems that need to be solved for HMMs to be useful in biological sequence analysis: the scoring, the detection, and the training problem. Efficient solution of these problems leads to many applications of HMMs in biology, including homology detection, for which profiles were originally designed, but also extremely rapid multiple sequence alignment.

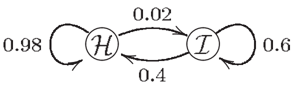
6.5 Further Reading

The tutorial by Rabiner is a very good place to learn more about hidden Markov models [205]. The textbook by Durbin and colleagues explains in detail how hidden Markov models are applied in biological sequence analysis [52].

6.6 Exercises and Software Demonstration

6.1. Write down the entries for position 21 in the profile shown in Figure 6.2.

6.2. Consider the Markov chain



(6.1)

describing the transitions between the states "ill" ($\mathcal{I}$) and "healthy" ($\mathcal{H}$). States can switch at daily intervals. Given that a person is ill now, what is the probability of observing the sequence $\mathcal{IIHHH}$ over the next five days?

6.3. What is the expected duration of illness and health according to HMM (6.1)?

6.4. Consider the HMM shown in Figure 6.8. The following six random numbers are drawn from a uniform distribution between 0 and 1: 0.222969, 0.731557, 0.583197, 0.547333, 0.821281, 0.163268. Use these numbers to generate the first three hidden and observed states of the model.

6.5. Fill in column 3 in the forward process matrix of Figure 6.10.

6.6. Fill in column 5 in the Viterbi matrix of Figure 6.11.

6.7. Given the DNA sequences $O_1 = \texttt{ATGA}$, $O_2 = \texttt{AGA}$, and $O_3 = \texttt{TGT}$ as well as the corresponding most likely state sequences $S_1 = m_1 m_2 m_3 m_4$, $S_2 = m_1 d_2 m_3 m_4$, and $S_3 = d_1 m_2 m_3 d_4 i_4$, what is the corresponding multiple sequence alignment?

6.8. Given the sequences $O_1 = \texttt{ATGA}$ and $O_2 = \texttt{ACA}$, as well as the corresponding most likely state sequences $S_1 = m_1 i_1 i_1 m_2$ and $S_2 = m_1 i_1 m_2$, is the induced sequence alignment unambiguous?

6.9. Work through the tutorial of the `bioinformer` program `HMM` → `Hidden Markov Model` given in Section A.3.1.

7

Gene Prediction

> *Rates of spontaneous mutation to recessive lethal and visible mutants in mammals are of the order of 10^{-6} to 10^{-5} per locus per generation. If there are 40,000 genes, the total rate of mutation to lethal or nonfunctional alleles would be between 4 and 40 percent per gamete. From this consideration alone, it is clear that there cannot be more than 40,000 genes.*
>
> *Jack Lester King and Thomas H. Jukes [145, p.794]*

7.1 What is a Gene?

The old question of "what is a gene" has been answered differently since their discovery by Mendel in 1866. Mendel thought of genes as mathematical objects and did not speculate on their material correlate. In 1927, Muller discovered that X-rays could induce mutations in *Drosophila* and this hinted that genes consisted of matter that interacted with X-rays. In addition, cytological studies correlated the action of genes with the structure and transport of chromosomes. However, chromosomes consist of both proteins and DNA and it was unclear which of the two constituted the genetic material. Certain was only that the molecule had to be able to self-replicate and to mutate. Because proteins are so much more complex than nucleic acids, it was often assumed that proteins carried the genetic information. In 1928, Griffith [91] observed that avirulent *Streptococcus pneumoniae* became virulent in the presence of heat-killed virulent cells. Therefore, the genetic material seemed to remain active even though its host was dead. In 1944, Avery and colleagues [13] repeated Griffith's transformation experiment but used DNA from virulent strains instead of whole heat-killed virulent cells and again obtained the virulent form of *S. pneumoniae* from avirulent bacteria. This established that Griffith's transforming principle was DNA. Nine years later Watson and Crick [256] published the structure of B-form DNA. The excitement about this structure was great because it fitted so well the expectations of what a gene should be: a molecule capable of instructing its own replication.

Today, we may think of a gene as a fragment of DNA that encodes a cellular molecule, protein or RNA. In procaryotes this usually includes the promoter region, which binds the RNA polymerase and is located at the 5' end of the gene. Eucaryotic genes may additionally be characterized by an intron/exon structure and signal sequences located at the 3' end of the gene or in introns. However, promoters and other signal sequences often remain unannotated. For example, Figure 7.1 shows the GenBank entry with accession number (`ACCESSION`) X78384 for the *Adh* gene of *Drosophila melanogaster* and its duplicate, the *Adh*$_{\mathrm{dup}}$ gene. The entry contains under the heading `FEATURES` annotations for introns, exons, the mRNA, and the protein coding sequence (CDS) but none for the promoter. Figure 7.2 gives a graphical representation of these features.

7.2 Computational Gene Finding

One aim of genome sequencing projects is to compile a catalog of all genes in a genome and there is a lot of interest in automating this process, called automatic sequence annotation. A number of programs have been published and are routinely used for this purpose.

Initially, programs were developed for procaryote sequences. In this case, gene finding by computational methods turned out to be relatively easy, because procaryotes have a high gene density and genes—with a few exceptions—do not have introns. For eucaryotes, the gene finding problem proved to be much more complicated.

In the years preceding the publication of the human genome, the total number of human genes was a hotly debated subject and bets on the size of the gene complement could be placed at a public website. Those numbers ranged from 20,000 to 200,000. When the two draft human genome sequences were published, the total gene number was estimated to be 31,000 [128] and 26,000 [252] genes in these papers. This is far fewer than previously expected by most scientists, but very close to the rough estimates from the 1950s [102] and 1960s [145], which were motivated by population genetic arguments.

Below we explain a few general principles of gene finding in eucaryotes and discuss why it still remains a difficult problem to exactly determine the number, location, and structure of genes in a genome.

The first programs for automatized gene prediction were created in the early 1980s [75]. Subsequently, gene prediction became a rapidly advancing field within bioinformatics. Today, different methods for this purpose are implemented in a large number of programs. Two classes of gene prediction methods can be distinguished, *ab initio* and *comparative*, sometimes also referred to as *intrinsic* and *extrinsic* (Fig. 7.3).

Ab initio gene prediction refers to those programs which search a single query sequence for certain signals—for example splice site signals—and compositional features, from which location and exon-intron structure of hypothetical genes are deduced. Comparative methods, on the other hand, build their gene predictions upon similarity of the query with other DNA or protein sequences and the differing degrees of similarity in functional and non-functional regions of the genome. Similarity is determined by an alignment procedure. Many of the programs which are in use for large scale genome annotation are hybrids of the two methods. Modern gene prediction programs can detect genes on either strand, forward or reverse, of a given query sequence (Fig. 7.4).

All gene prediction programs aim to decipher the exon-intron structure, i.e the exact location of splice sites, and translation start and stop codons of the coding

```
LOCUS       DMADH                  4761 bp    DNA     linear   INV 09-MAY-1997
DEFINITION  D.melanogaster Adh and Adh-dup genes.
ACCESSION   X78384
VERSION     X78384.1  GI:483718
KEYWORDS    ADH gene; Adh-dup gene; polymorphism.
SOURCE      Drosophila melanogaster.
  ORGANISM  Drosophila melanogaster
            Eukaryota; Metazoa; Arthropoda; Hexapoda; Insecta; Pterygota;
            Neoptera; Endopterygota; Diptera; Brachycera; Muscomorpha;
            Ephydroidea; Drosophilidae; Drosophila.
REFERENCE   1  (bases 1 to 4761)
  AUTHORS   Kreitman,M. and Hudson,R.R.
  TITLE     Inferring the evolutionary histories of the Adh and Adh-dup loci in
            Drosophila melanogaster from patterns of polymorphism and
            divergence
  JOURNAL   Genetics 127 (3), 565-582 (1991)
  MEDLINE   91200630
   PUBMED   1673107
COMMENT     See paper for sequence comparison to D. simulans.
FEATURES             Location/Qualifiers
     source          1..4761
                     /organism="Drosophila melanogaster"
                     /db_xref="taxon:7227"
                     /chromosome="2"
                     /note="for sequence comparison with D.simulans see paper"
     gene            join(1244..1330,1985..2119,2185..2589,2660..3100)
                     /gene="Adh"
     mRNA            join(1244..1330,1985..2119,2185..2589,2660..3100)
                     /gene="Adh"
                     /note="distal transcript
                     for sequence comparison with D.simulans see paper"
     exon            1244..1330
                     /gene="Adh"
                     /number=1
     intron          1331..1950
                     /gene="Adh"
                     /number=1
     exon            1985..2119
                     /gene="Adh"
                     /number=2
     CDS             join(2021..2119,2185..2589,2660..2926)
                     /gene="Adh"
                     /note="unnamed protein product"
                     /codon_start=1
                     /protein_id="CAA55151.1"
                     /db_xref="GI:2077989"
                     /db_xref="FLYBASE:FBgn0000055"
                     /db_xref="SWISS-PROT:P00334"
                     /translation="MSFTLTNKNVIFVAGLGGIGLDTSKELLKRDLKNLVILDRIENP
                     AAIAELKAINPKVTVTFYPYDVTVPIAETTKLLKTIFAQLKTVDVLINGAGILDDHQI
                     ERTIAVNYTGLVNTTTAILDFWDKRKGGPGGIICNIGSVTGFNAIYQVPVYSGTKAAV
                     VNFTSSLAKLAPITGVTAYTVNPGITRTTLVHKFNSWLDVEPQVAEKLLAHPTQPSLA
                     CAENFVKAIELNQNGAIWKLDLGTLEAIQWTKHWDSGI"
     intron          2120..2184
                     /gene="Adh"
                     /number=2
     exon            2185..2589
                     /gene="Adh"
                     /number=3
     intron          2590..2659
                     /gene="Adh"
```

```
                     /number=3
     exon            2660..3100
                     /gene="Adh"
                     /number=4
     gene            3226..4761
                     /gene="Adh-dup"
     mRNA            join(<3226..3321,3748..4152,4204..4761)
                     /gene="Adh-dup"
                     /note="for sequence comparison with D.simulans see paper"
     CDS             join(3226..3321,3748..4152,4204..4521)
                     /gene="Adh-dup"
                     /note="for sequence comparison with D.simulans see paper"
                     /codon_start=1
                     /protein_id="CAA55152.1"
                     /db_xref="GI:483719"
                     /db_xref="FLYBASE:FBgn0000056"
                     /db_xref="SWISS-PROT:P91615"
                     /translation="MFDLTGKHVCYVADCGGIALETSKVLMTKNIAKLAILQSTENPQ
                     AIAQLQSIKPSTQIFFWTYDVTMAREDMKKYFDEVMVQMDYIDVLINGATLCDENNID
                     ATINTNLTGMMNTVATVLPYMDRKIGGTGGLIVNVTSVIGLDPSPVFCAYSASKFGVI
                     GFTRSLADPLYYSQNGVAVMAVCCGPTRVFVDRELKAFLEYGQSFADRLRRAPCQSTS
                     VCGQNIVNAIERSENGQIWIADKGGLELVKLHWYWHMADQFVHYMQSNDEEDQD"
     exon            <3226..3321
                     /gene="Adh-dup"
                     /number=1
     intron          3322..3747
                     /gene="Adh-dup"
                     /number=1
     exon            3748..4152
                     /gene="Adh-dup"
                     /number=2
     intron          4153..4203
                     /gene="Adh-dup"
                     /number=2
     exon            4204..4761
                     /gene="Adh-dup"
                     /number=3
BASE COUNT     1417 a   1007 c    989 g   1348 t
ORIGIN
        1 tgtattttcc aattaggtga tagaacttgt gtgcacacac acatatagtt ctatatcaac
       61 aaacaggttt aagttttatg caaattgaaa gcttatttct tccgcatgct tatctctttc
      121 cttctcatca tttgtatgca aaaaatacat atgaatttgc agtagcctcc tcccacatca
      181 tatttaacgc cctatattca aaatttgctc aagaaaatat ttgaaccaaa ttgattttta
      241 gtcaattagt ttttaagtaa ttaagtggag taaacatata caattttatt cttaccaaac
      301 acatatactc atatatttttg aataaataaa taaacaaata tatataaaat ctacgaaatt
      361 ggcaaacaaa ttttaaagca ttatagtatt gccgatttaa ttaatataat taaataatat
      421 gtacatgtat taatcttgtg tgcgagcatg ggttaaatct agctgcattc gaaaccgcta
      ... .......... .......... .......... .......... .......... ..........
     2101 gctcaagcgc gatctgaagg taactatgcg atgcccacag gctccatgca gcgatggagg
     2161 ttaatctcgt gtattcaatc ctagaacctg gtgatcctcg accgcattga gaacccggct
     2221 gccattgccg agctgaaggc aatcaatcca aaggtgaccg tcaccttcta cccctatgat
     2281 gtgaccgtgc ccattgccga gaccaccaag ctgctgaaga ccatcttcgc ccagctgaag
     2341 accgtcgatg tcctgatcaa cggagctggt atcctggacg atcaccagat cgagcgcacc
     2401 attgccgtca actacactgg cctggtcaac accacgacgg ccattctgga cttctgggac
     2461 aagcgcaagg gcggtcccgg tggtatcatc tgcaacattg gatccgtcac tggattcaat
     2521 gccatctacc aggtgcccgt ctactccggc accaaggccg ccgtggtcaa cttcaccagc
     2581 tccctggcgg taagttgatc aaaggaaacg caaagttttc aagaaaaaac aaaactattt
     2641 gattttataa cacctttaga aactggcccc cattaccggc gtgaccgctt acaccgtgaa
     2701 ccccggcatc acccgcacca ccctggtgca caagttcaac tcctggttgg atgttgagcc
     2761 ccaggttgct gagaagctcc tggctcatcc cacccagcca tcgttggcct gcgccgagaa
     2821 gttgtcagtt gcagttcagc agacgggcta acgagtactt gcatctcttc ..........
```

Fig. 7.1. GenBank entry X78384 with abridged DNA sequence (indicated by dots in the `ORIGIN` feature).

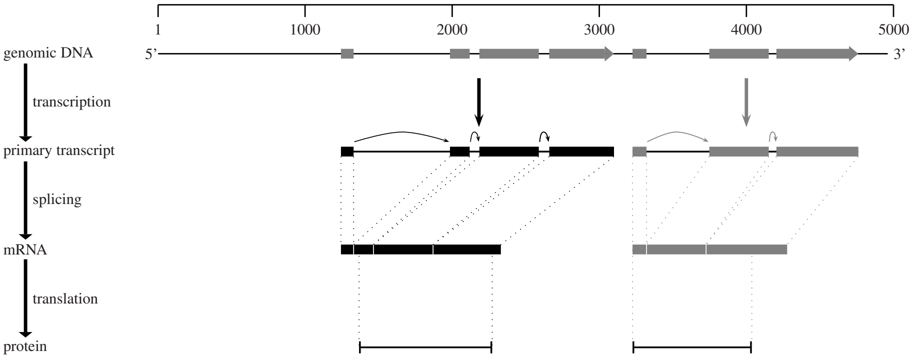

1
1000
2000
3000
4000
5000
genomic DNA
5'
3'
transcription
primary transcript
splicing
mRNA
translation
protein

Fig. 7.2. Gene structure at the *Adh* locus in *D. melanogaster*, as annotated in GenBank entry X78384. Two genes are shown, *Adh* in black and its duplicate *Adh*$_{\text{dup}}$ in gray. Boxes denote exons, lines in the primary transcript denote introns; notice that translation of *Adh* starts in its second exon; in *Adh*$_{\text{dup}}$ the mRNA is only partially annotated and the translation start is at the beginning of the first annotated exon.

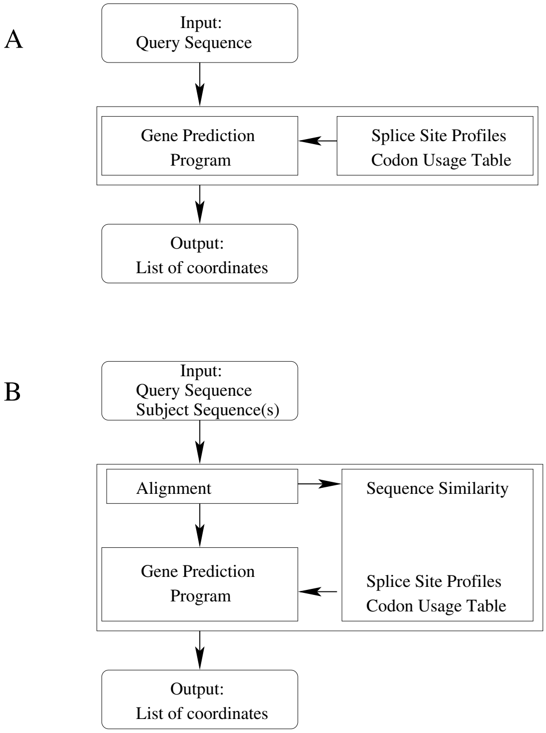

A
Input:
Query Sequence
Gene Prediction
Program
Splice Site Profiles
Codon Usage Table
Output:
List of coordinates
B
Input:
Query Sequence
Subject Sequence(s)
Alignment
Sequence Similarity
Gene Prediction
Program
Splice Site Profiles
Codon Usage Table
Output:
List of coordinates

Fig. 7.3. Schematic flow chart of gene prediction algorithms. **A**: *Ab initio*; **B**: comparative.

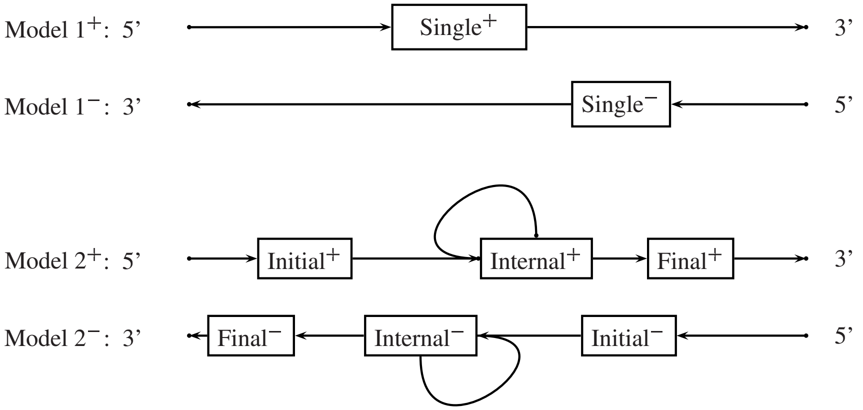


Fig. 7.4. There are two principal gene models: single-exon (*Model 1*) and multi-exon (*Model 2*) genes. For a given query sequence there may be genes on the positive (forward) strand and on the negative (backward) strand. To decipher genes on the backward strand, the reverse complement of the query sequence needs to be considered. Multi-exon genes can have any number of internal exons, indicated by the back-looping arrow.

sequence (CDS) of protein coding genes. Some programs additionally try to identify the exon-intron structure in the untranslated 5' and 3' regions of genes. And still others also predict further upstream or downstream sequence signals, such as putative binding sites for transcription factors or polyadenylation sites.

7.3 Measuring the Accuracy of Gene Predictions

A principal concern with gene prediction programs is their accuracy. In order to quantify this, a standardized accuracy measure is needed. Such a measure has been defined by Burset and Guigó [33]. To evaluate the accuracy of a gene prediction program on a test sequence, the gene structure predicted by the program is compared with the actual gene structure of the sequence as it has been established with the help of an experimentally validated mRNA. The underlying assumption is that there exists a well-defined, true gene structure which is not confounded by alternative splice variants. In such a situation the accuracy of gene prediction can be evaluated at three different levels of resolution, i.e. nucleotide, exon, and gene. These three levels offer complementary views of the accuracy of the program. At each level, there are two basic measures: *sensitivity* (S_n) and *specificity* (S_p). Sensitivity is the proportion of correctly predicted elements (nucleotides, exons, or genes) among the true elements. In contrast, specificity is the proportion of correctly predicted elements, among all elements predicted. Consider the prediction at the nucleotide level (N) shown in Figure 7.5. In this case, sensitivity is

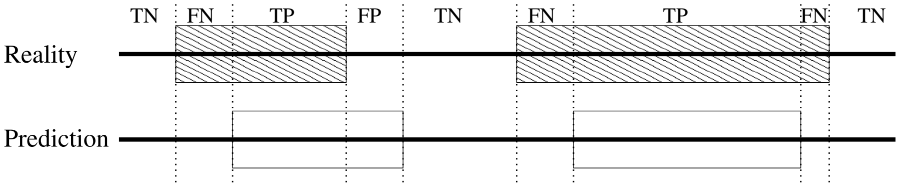


Fig. 7.5. Measuring gene prediction accuracy at the level of nucleotides is based on the number of true positive (TP), true negative (TN), false positive (FP), and false negative (FN) nucleotides.

$$S_n(N) = \frac{\mathrm{TP}(N)}{\mathrm{TP}(N) + \mathrm{FN}(N)}$$

and specificity is

$$S_p(N) = \frac{\mathrm{TP}(N)}{\mathrm{TP}(N) + \mathrm{FP}(N)}.$$

Both sensitivity and specificity take values from 0 to 1. Perfect prediction is characterized by both measures being equal to 1. Neither S_n nor S_p alone constitute good measures of global accuracy, since one can have high sensitivity with little specificity and *vice versa*. It is desirable to use a single scalar value to summarize both of them. In the gene finding literature, the preferred measure on the nucleotide level is the correlation coefficient. It is defined as

$$\mathrm{CC} = \frac{\mathrm{TP}(N)\cdot\mathrm{TN}(N) - \mathrm{FP}(N)\cdot\mathrm{FN}(N)}{\sqrt{(\mathrm{TP}(N)+\mathrm{FP}(N))(\mathrm{TN}(N)+\mathrm{FN}(N))(\mathrm{TN}(N)+\mathrm{FP}(N))(\mathrm{TP}(N)+\mathrm{FN}(N))}}. \tag{7.1}$$

CC ranges from -1 to 1, with 1 corresponding to perfect prediction and -1 to a prediction in which each coding nucleotide is classified non-coding and *vice versa*. At the exon level (E), an exon is considered a true positive exon only if the predicted exon is identical to the true one, in particular both 5' and 3' exon boundaries have to be correct. A predicted exon is considered to be false positive, if it has no overlap with any real exon. Sensitivity and specificity are the proportion of true positive exons among all true exons and among all predicted exons, respectively. A summary measure on the exon level is simply the mean of sensitivity and specificity. At the gene level (G), a gene is considered to be a true positive (TP(G)) if all of the coding exons are identified, every intron-exon boundary is correct, and all of the exons are included in the proper gene. A gene is a false negative (FN(G)), or missed, if none of its exons is overlapped by any predicted gene. Conversely, a predicted gene is a false positive (FP(G)), or wrong, if none of its exons is overlapped by any real gene. It is a well known problem that *ab initio* gene finders predict the initial and terminal exons of a gene very poorly. This often leads to so-called chimeric predictions, one predicted gene encompasses more than one real gene, or to split predictions, one real gene is split in multiple predicted genes. Two measures, split genes, SG, and joined genes, JG, have been proposed to account for these tendencies [206]. SG is the total number of predicted genes overlapping real genes, divided by the number of genes that were split. Similarly, JG is the total number of real genes that overlap predicted genes, divided by the number of predicted genes that were joined. Note, that SG and JG may be larger than 1. The measures are summarized in Table 7.1.

Table 7.1. Basic definitions of accuracy measures [97].

	Nucleotide	Exon	Gene
Sensitivity S_n	proportion of coding nucleotides correctly predicted as coding	proportion of correctly predicted exons among actual exons	proportion of completely correctly predicted genes among actual genes
Specificity S_p	proportion of nucleotides predicted as coding and which are actually coding	proportion of correctly predicted exons among all predicted exons	proportion of completely correctly predicted genes among all predicted genes

In practice, most programs achieve quite high values of gene prediction accuracy at the nucleotide level, but much lower values at the level of exons, and very poor values at the level of genes (Table 7.2).

Table 7.2. Accuracy on the nucleotide, exon, and gene levels of several gene finders used in the GASP experiment [206] in which 2.9 Mb of genomic sequence from the *Adh* region of *D. melanogaster* were annotated. For an explanation of S_n, S_p, FN, FP, SG, and JG see text. Rows correspond to the results obtained with (1) `fgenes` [213], (2) `GeneID` [96], (3) `genie` [157], (4) `genie` EST-version [207], (5) `grail` [248], (6) `hmmgene` [154], and (7) `magpie` [83].

	Nucleotide		Exon				Gene					
	S_n	S_p	S_n	S_p	FN(E)	FP(E)	S_n	S_p	FN(G)	FP(G)	SG	JG
(1)	0.89	0.77	0.65	0.49	0.10	0.32	0.30	0.27	0.09	0.24	1.10	1.06
(2)	0.86	0.83	0.58	0.34	0.21	0.47	0.26	0.10	0.14	0.30	1.06	1.11
(3)	0.96	0.92	0.70	0.57	0.08	0.17	0.40	0.29	0.05	0.11	1.17	1.08
(4)	0.97	0.91	0.77	0.55	0.05	0.20	0.44	0.28	0.05	0.13	1.15	1.09
(5)	0.81	0.86	0.42	0.41	0.24	0.29	0.14	0.12	0.16	0.24	1.23	1.08
(6)	0.97	0.91	0.68	0.53	0.05	0.20	0.35	0.30	0.07	0.15	1.04	1.12
(7)	0.96	0.63	0.63	0.41	0.12	0.50	0.33	0.21	0.05	0.55	1.22	1.06

7.4 *Ab initio* Methods: Searching for Signals and Content

The ideal gene finding program would perfectly mimic the cell's transcription, splicing and translation machinery. No such program exists, but a number of biologically important sequence signals are exploited in existing algorithms.

Transcription Signals

There are three principal signals:

1. Transcription start site (TSS): Characterized by the CAP signal, a single purine (`A`/`G`).
2. TATA-box: A/T-rich region about 30 bp upstream from the TSS.
3. Transcription termination: characterized by the so-called polyadenylation signal, a consensus `AATAAA` hexamer and a further, degenerate signal 20/30 bp downstream from the polyadenylation signal.

Unfortunately, only 70% of human promoters actually contain these core signal sequences. Moreover, the `AATAAA` polyadenylation signal is absent from 50% of all genes. Hence, it is hard to determine the beginning and the end of a gene.

Translational Signals

Two signals are important here:

1. The sequence motif `gccaccAUGG` is the optimal context for initiation of translation in vertebrate mRNA. This is sometimes referred to as the 'Kozak signal'. The corresponding signal sequence in DNA is `gccaccATGG`.
2. Termination codon: any one of the three stop codons `TGA`, `TAA`, and `TAG` should occur at the end of the coding sequence (CDS), but not in the internal part. An exception is the situation in which, due to alternative splicing, a premature stop codon may lead in some transcript variants to an incomplete peptide. Furthermore, it is now known that, in certain very rare contexts, the triplet `TGA` can code for selenocysteine, which is sometimes referred to as the 21-st amino acid [24, 156].

Splicing Signals

Introns in pre-mRNA transcribed from nuclear genes are excised by spliceosomes. These are large ribonucleoprotein complexes that recognize three kinds of sites:

1. The 5' end of an intron, the so-called donor site, characterized by the dinucleotide `GT`.
2. The 3' end of an intron, the so-called acceptor site, characterized by the dinucleotide `AG` and an upstream poly-pyrimidine tract.
3. The branch point, located close to the 3' end of the intron and upstream of its poly-pyrimidine tract. It is a degenerate motif [105] always containing the nucleotide `A`. The mammalian consensus motif is the heptamer `YNYTRAY` [228, 199].

The rules about the donor and acceptor sites are almost universal with a few exceptions. The most frequent non-standard donor dinucleotide is `GC`, which occurs in mammals in less than 1% of the exons [34]. Whether a splice-site is used or not may also be influenced by additional exonic and intronic signals, known as exonic and intronic splice-enhancers, which are located some distance from the splice junctions.

Programs which use such signals for gene prediction are also said to perform a *search by signal*.

As illustrated in Figure 7.6, the reading frame is the offset at the beginning of the exon when the DNA sequence is translated into a sequence of amino acids. It can therefore take one of three possible values: 0, 1, or 2. Initial and single exons always have reading frame 0. The reading frame of an internal or terminal exon depends on length and reading frame of the previous exon. It can be recursively determined by

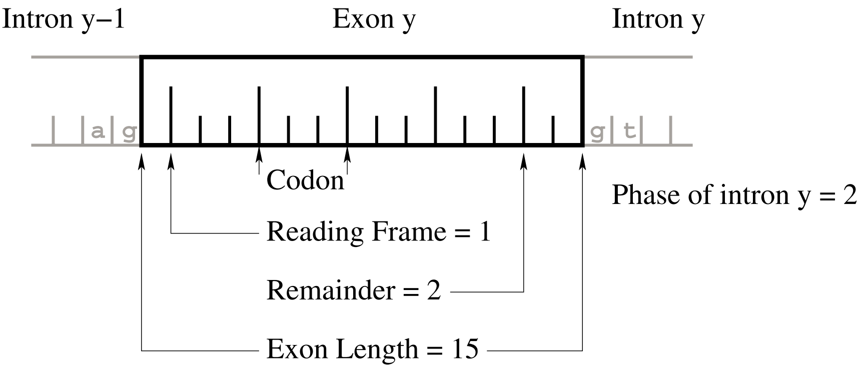


Fig. 7.6. Definition of reading frame, remainder, and intron phase.

$$\begin{aligned}\text{frame}(1) &= 0\\ \text{frame}(y) &= (3 + \text{frame}(y-1) - \text{length}(y-1)\%3)\%3\,, \end{aligned} \tag{7.2}$$

where $y > 1$ denotes the y-th exon of a gene. Thus, one obtains the following scheme for the frame of exon number y:

$$\begin{array}{r|ccc} \text{length}(y-1)\%3 \rightarrow & 0 & 1 & 2\\ \text{frame}(y-1) \quad \downarrow & & & \\ \hline 0 & 0 & 2 & 1\\ 1 & 1 & 0 & 2\\ 2 & 2 & 1 & 0 \end{array}$$

Equivalently, introns can be classified into three different types. These are often called introns of *phase* 0, 1, or 2 (Fig. 7.6). From an intron phase the reading frame can be uniquely determined and *vice versa*. Still another, equivalent, characteristic is the *remainder* of an exon. As shown in Figure 7.6, this is the surplus of nucleotides (0, 1, or 2) with respect to the end of the last complete codon at the 3’ end of an exon.

Table 7.3. Exon types in eucaryotic protein-coding genes and their standard splice signals. Capital letters: exonic signal; lower case letters: intronic signal.

type	3' signal		5' signal		frame	remainder
first	initiation codon	`ATG`	donor site	`gt`	0	0,1,2
internal	acceptor site	`ag`	donor site	`gt`	0,1,2	0,1,2
terminal	acceptor site	`ag`	stop codon	`TAR,TGA`	0,1,2	0
single	initiation codon	`ATG`	stop codon	`TAR,TGA`	0	0

Most programs concentrate on the prediction of protein-coding exons and distinguish four types of exons: first, internal, terminal, and single exons. Table 7.3 summarizes the sequence characteristics of these four exon types.

7.4.1 Codon Usage

Another important measure which informs prediction algorithms about coding and non-coding parts of the query sequence is the frequency with which alternative codons appear in the query sequence. In contrast to signal searches, programs which use this kind of information are said to perform a *search by content*. Different amino acids have different degrees of degeneracy in the genetic code (cf. Table C.4). There are unique codons and 2-, 3-, 4-, and 6-fold degenerate codons. For instance, the nucleotide triplets `GGA`, `GGC`, `GGG`, and `GGT` all code for the same amino acid glycine. Thus, this set of codons is 4-fold degenerate. However, degenerate codons generally are not used randomly but a particular codon is preferred. This property is called codon bias. While being particularly strong in highly expressed genes, it is also characteristic of a given species. Codon bias is caused by differential abundance of the respective tRNAs. Figure 7.7 shows that the possible codons for a given amino acid are used with different relative frequencies in mammals and insects (human/*Drosophila* comparison), but that there are also differences within mammals (human/mouse comparison). This explains why gene prediction programs need to be optimized for each species separately. This process is also called parameter-training.

7.4.2 Finding Splice Sites with a Sequence Profile

A common way to identify sequence signals is via a position weight matrix (PWM). To illustrate its construction and application we build a "primitive donor finder", called PDF. It is known that nucleotides around the `GT` motif are not randomly distributed. To see this, we select from a database such as GenBank a set of experimentally verified donors and extract the donor dinucleotide plus 3 nucleotides upstream and 4 nucleotides downstream. For instance, GenBank entries U04239 and M61127 represent two genomic sequences from *D. melanogaster* with one multi-exon gene each. From these entries the eight sequence fragments shown in Table 7.4 are obtained. Counting the number of occurrences of each nucleotide at each position in the alignment we derive the matrix of relative frequencies shown in Figure 7.8.

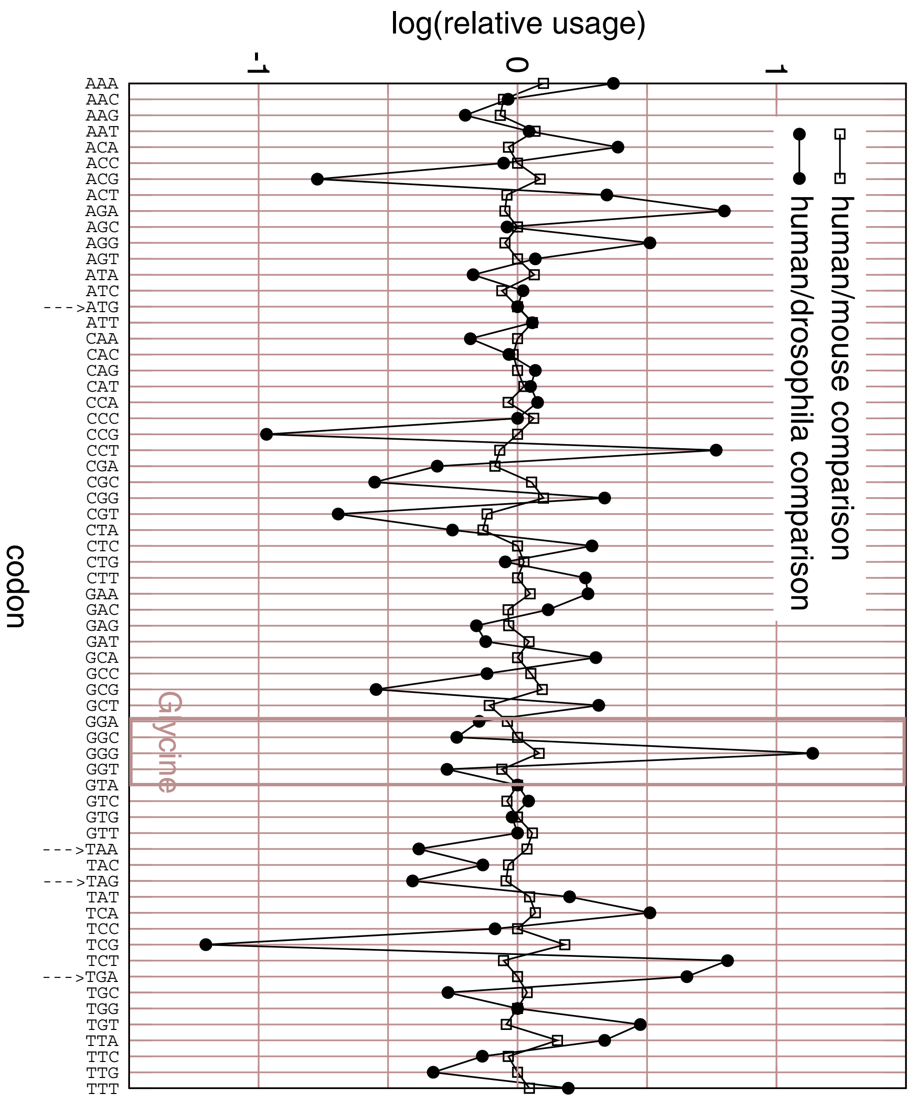


Fig. 7.7. Differential codon usage in human, mouse, and *Drosophila*. Vertical axis: 64 codons; horizontal axis: $\log(x/y)$, where x and y are the relative codon frequencies for a given amino acid in human/mouse and human/*Drosophila* comparisons. Non-degenerate codons are necessarily unbiased, for instance the codon `ATG`.

Table 7.4. Alignment of eight nonamers comprising the splice donors from two genomic sequences of *D. melanogaster*, GenBank entries U04239 and M61127. The donor dinucleotide is displayed in bold. Nucleotides within exons are capitalized.

fragment	accession number	5' end of intron	position	sequence
f_1	U04239	donor 1	868-876	CTG**gt**gagt
f_2	U04239	donor 2	992-1000	GGG**gt**aagt
f_3	U04239	donor 3	1481-1489	CGG**gt**aagt
f_4	U04239	donor 4	2356-2364	AGG**gt**aagt
f_5	U04239	donor 5	2669-2677	AAA**gt**aagt
f_6	U04239	donor 6	3291-3299	TAG**gt**aact
f_7	M61127	donor 1	226-234	TGG**gt**ttgt
f_8	M61127	donor 2	539-547	TAG**gt**gagt

	-3	-2	-1	0	1	2	3	4	5
A	2/8	3/8	1/8	0/8	0/8	5/8	7/8	0/8	0/8
C	2/8	0/8	0/8	0/8	0/8	0/8	0/8	1/8	0/8
G	1/8	4/8	7/8	8/8	0/8	2/8	0/8	7/8	0/8
T	3/8	1/8	0/8	0/8	8/8	1/8	1/8	0/8	8/8

Fig. 7.8. Frequency matrix of each nucleotide at each of the nine positions in the alignment in Table 7.4. The positions of the donor dinucleotide are set to 0 and 1.

How do the observed nucleotide frequencies from donor sites compare to nucleotide frequencies from random nonamers? To simplify matters, let us assume that the four nucleotides occur with equal frequencies in the genome of *D. melanogaster*. We then determine the log-likelihood of each nucleotide at each position by taking the logarithm of the ratio of observed and expected frequencies, $\log_2(O/E)$, and obtain the log-likelihood, or position weight matrix (PWM), shown in Figure 7.9. The PWM is said to be of Markov order 0, if the frequencies at all sites are evaluated independently, as done here. For instance, consider the entry 0.58 at position 2 for nucleotide `A`. It is computed as

$$\log_2\left(\frac{3/8}{1/4}\right) \approx 0.58\,.$$

As described in Chapter 6, a query sequence can be scanned with a PWM profile and a score $s(y)$ can be assigned to each position y of the query. High scoring donor candidates are those positions where the score exceeds some (positive) threshold. When choosing the logarithm with base two, as done here, the log-likelihoods have an information-theoretical interpretation. Consider a set of random oligomers of a fixed length, say nine. The "uncertainty" [224] at each site x in this set of oligomers is given by

$$H(x) = -\sum_{i=A,C,G,T} f(i,x)\log_2 f(i,x)\,,$$

	-3	-2	-1	0	1	2	3	4	5
A	0	0.58	-1	$-\infty$	$-\infty$	1.32	1.81	$-\infty$	$-\infty$
C	0	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	-1	$-\infty$
G	-1	1	1.81	2	$-\infty$	0	$-\infty$	1.81	$-\infty$
T	0.58	-1	$-\infty$	$-\infty$	2	-1	-1	$-\infty$	2

Fig. 7.9. Position weight matrix for the primitive donor finder (see text).

where $f(i,x)$ is the relative frequency of nucleotide i at position x. When the background nucleotide frequencies are all equal, $f(i,x) = 1/4$, uncertainty is $H(x) = 2$. The "information content" of a site x in a profile such as Figure 7.9 is the "decrease in uncertainty", $R(x)$, and is given by

$$R(x) = 2 - H(x)\,.$$

For the nine positions in the donor profile we derive the information content given in Table 7.5.

Table 7.5. Information content at the individual sites in the donor profile from Figure 7.9. For numerical evaluation of the information content we take $\lim_{\chi \to f(i,x)} \chi \log_2(\chi)$ in order to avoid undefined values when the relative frequency is equal to 0.

position x	information content $R(x)$
-3	0.09
-2	0.59
-1	1.46
0	2.00
1	2.00
2	0.70
3	1.46
4	1.46
5	2.00

The information content of a profile can be visualized graphically as a so-called *sequence logo* [219]. Figure 7.10A displays the sequence logo for the donor profile of the primitive donor finder. At each position x letters `A`, `C`, `G`, `T` are stacked on top of each other. The height of a letter, i, is given by the product $f(i,x)R(x)$; the combined height is then equal to the information content of x.

We now use the primitive donor finder to search for putative donor sites in a genomic sequence. As a test sequence we choose the *Adh* locus from *D. melanogaster* (cf. Figure 7.2). We find three putative donor sites, indicated by arrows in Figure 7.11. The donor scores in Figure 7.11 are simply the sum of the respective entries in the profile weight matrix. Only donors with positive scores are shown. By

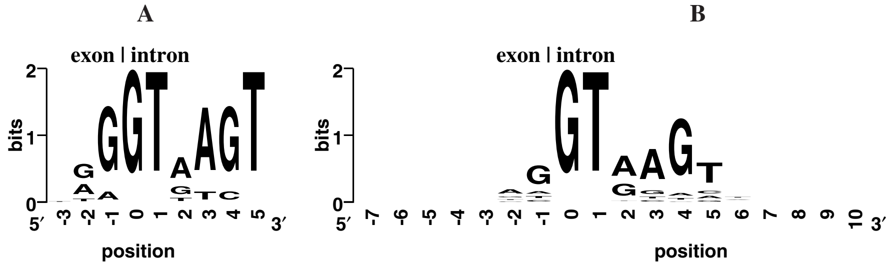


Fig. 7.10. A: Sequence logo of the eight donor sequences used for the primitive donor finder shown in Table 7.4. **B**: Sequence logo built from 757 *Drosophila* donors available at `http://www.fruitfly.org/GASP1/data/data.html`. X-axis: sequence position around donor dinucleotide `GT` (at positions 0 and 1); Y-axis: information content at each position in bits. Note that in **B** the number of informative positions around a *Drosophila* donor site is limited to about five nucleotides downstream of `GT` and two nucleotides upstream.

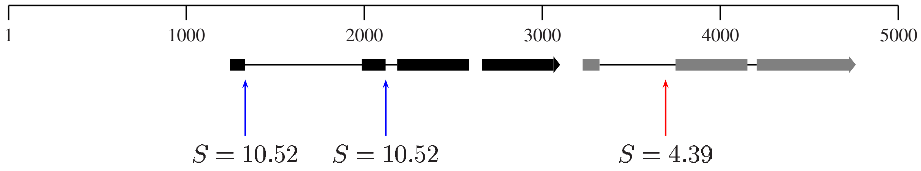


Fig. 7.11. Applying the primitive donor finder PDF to GenBank entry X78384. The vertical arrows indicate positions of putative donors with a positive score.

comparing this result to the mRNA feature in the corresponding GenBank entry in Figure 7.1, we see that the first two reported donor sites are true donors of the *Adh* gene and that the third predicted donor site is a false positive: it does not correspond to any of the annotated donor sites. Furthermore, the PDF tool fails to recover the third donor of the *Adh* gene and both donors of the Adh_{dup} gene (cf. the annotation in Figure 7.1). Thus, there are three false negative predictions, and the PDF has a sensitivity of

$$\frac{2}{5} = 0.40$$

and a specificity of

$$\frac{2}{3} = 0.66.$$

Clearly, these values are not satisfactory. A strategy to improve the low accuracy of our PDF tool would be to compile a much larger training set of verified donors in order to build a better tuned position weight matrix. One such set, for instance, has been compiled during the GASP [206] experiment and can be accessed through their website. The sequence logo obtained with this set is displayed in Figure 7.10B. There are additional parameters which should be optimized. For instance, the score threshold below which a putative donor site is discarded, may be chosen to be different from zero, as was done here. The optimization criterion is to maximize on a well-defined training set some function of sensitivity and specificity, $f(S_n; S_p)$, for instance the average $(S_n + S_p)/2$.

7.4.3 Exon Chaining

In a similar manner to the donor search, genomic sequences may be scanned with the help of position weight matrices for acceptor sites, translation start, and stop signals. These scans can be combined to yield candidate exons of the four types: single, initial, internal, and terminal. The next step in predicting gene models with multiple exons is to select suitable exons from the lists of candidates. This problem can be solved by chaining of compatible exons [95]. Consider for example the "exons" shown in Figure 7.12. Each exon has a score, indicated by its position along the Y-axis, and an identifier, indicated by the number written on top. What we are looking for is the combination of non-overlapping exon candidates that yield the maximum score. In our example this combination happens to consist of intervals 9, 7, 3, and 0, which are shown in bold; their combined score is 31. The chaining algorithm shown in Figure 7.13 can be used to find such an optimal combination of exon candidates. Chaining is based on dynamic programming and starts from a list of n exon candidates, e. The algorithm consists of the standard two phases of dynamic programming: (i) a forward phase in which a table of size n is filled in to find the score of the optimal solution and (ii) a traceback phase in which the intervals belonging to the optimal solution are retrieved.

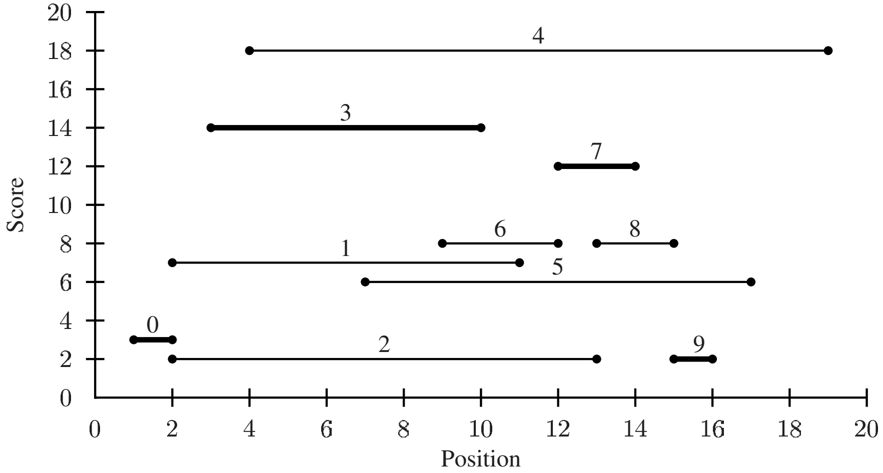


Fig. 7.12. The chaining problem consists of finding a path of non-intersecting intervals such that the combined score of the intervals which are included in the path is maximized (bold lines).

```
Require: $e$ //array of $n$ intervals
Require: $b$ //sorted array of the $2n$ borders of the $n$ intervals
  //Forward Algorithm
  $s \leftarrow 0$
  for all $0 \leq i < n$ do
    $p_i \leftarrow -1$ //initialize backpointers to "NULL"
    $v_i \leftarrow$ score of $e_i$
  $l \leftarrow -1$ //initialize index to last element added to optimal chain
  for $i \leftarrow 0$ to $2n - 1$ do
    if $b_i$ is left border of interval $e_j$ then
      $v_j \leftarrow v_j + s$
      $p_j \leftarrow l$
    else
      if $v_j > s$ then
        $s \leftarrow v_j$
        $l \leftarrow j$
  output $s$
  //Traceback
  while $l \neq -1$ do
    output $e_i$
    $l \leftarrow p_l$
```

Fig. 7.13. Exon chaining algorithm.

Apart from the list of exons, we also assume as given a corresponding list of exon borders, b, sorted in ascending order. Each border contains a reference to its interval of origin. Table 7.6 shows the border table corresponding to our example set of exons. The algorithm starts by initializing four variables:

1. s: the current score of the growing chain is set to zero;
2. v: the array of cumulative scores is set to the original interval scores;
3. p: the array of back pointer indices to the previous element in the chain is set to "NULL";
4. l: the index of the last element added to the chain is also set to "NULL".

Figure 7.14A shows the initialized variable v. After initialization, the algorithm proceeds along the list of borders starting at the left-most element. Say, the algorithm is currently dealing with border b_i, which refers to some interval e_j. If b_i is a left border of interval e_j, two operations are carried out:

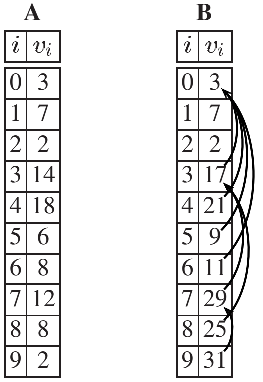
1. v_j is set to the score that the chain would have, if e_j was added to it; and
2. p_j is set to the last element that was added to the chain.

If, on the other hand, b_i is a right border, we need to decide whether or not to add e_j to the chain. Addition is executed if the score of the chain including e_j, v_j, is greater than s, the score of the current chain. In that case s is updated to the new score and l to the index of the current exon, j. If e_j is not added to the chain, nothing is done. Figure 7.14B shows v and the back pointers at the end of the forward phase of the algorithm. Next, the traceback starts from l, which in our case is 9. By following the back pointers, we get the exons shown in bold in Figure 7.12.

Table 7.6. Sorted list of borders of the intervals shown in Figure 7.12.

#	Position	Interval	Side
0	1	0	l
1	2	2	l
2	2	1	l
3	2	0	r
4	3	3	l
5	4	4	l
6	7	5	l
7	9	6	l
8	10	3	r
9	11	1	r
10	12	7	l
11	12	6	r
12	13	8	l
13	13	2	r
14	14	7	r
15	15	9	l
16	15	8	r
17	16	9	r
18	17	5	r
19	19	4	r

The chaining algorithm as described so far still needs to be refined by incorporating more of the biological reality it is designed to model. For a start, only those exons which do not disrupt the reading frame can be included in a chain; furthermore, it is attempted to build complete chains, which consist of exactly one initial exon, an arbitrary number of internal exons, and one terminal exon. This procedure resembles very much the way in which tiles in a domino game are laid out.



A

i	v_i
0	3
1	7
2	2
3	14
4	18
5	6
6	8
7	12
8	8
9	2

B

i	v_i
0	3
1	7
2	2
3	17
4	21
5	9
6	11
7	29
8	25
9	31

Fig. 7.14. Scores and back pointers for chaining the exons shown in Figure 7.12. **A**: Initialized array; **B**: filled in array.

In practice, before the chaining step is performed, additional requirements are imposed to reduce the number of candidate exons. For instance, a coding exon cannot contain internal stop codons and its length must fall within some pre-set range.

Furthermore, the distance between neighboring exons in multi-exon genes may not exceed an upper threshold. On the other hand, such requirements may themselves become a source of errors. For example, the intron length distribution is highly variable among species and giant introns are known to exist in most vertebrate genomes. Given a conservative threshold regime, very long introns might be missed, possibly resulting in the erroneous prediction of split genes.

In addition to these structural properties, measures of sequence composition are also used to better discriminate between coding and non-coding sequences. One such measure is codon bias. Furthermore, neighboring nucleotides are not independent of each other. Therefore, nucleotide composition, in particular of coding regions, is not adequately described by a Markov model of order zero. Experiments have shown that it is a good strategy to consider instead the hexamer composition of a sequence. A Markov model of order k accounts for dependencies at the level of $k{+}1$-mers. Hence, a fifth order Markov model is sensitive to dependencies at the level of hexamers. In a homogeneous model all positions are treated equally. In a non-homogeneous model one can, for instance, account for the triplet structure of typical coding regions. The gene prediction program `GenScan` [30] uses a homogeneous fifth order Markov model for non-coding regions and a three-periodic fifth order Markov model for coding regions. The parameters for this model are obtained from a training set with known coding and non-coding regions. The switching between the coding and the non-coding state is what makes the model a hidden Markov model. Hidden, because only the emitted nucleotide sequences are directly observable, while the underlying state (intron, exon, or intergenic) of the model remains hidden. `GenScan` switches between states according to a Markov chain, but in addition the duration of each state is influenced by empirical probability distributions of intron and exon lengths.

7.5 Comparative Methods

7.5.1 General Remarks

Instead of relying only on sequence signals and compositional features deduced from the query sequence itself, comparative methods also use information obtained from additional, homologous sequences. There are three types to be distinguished: (i) methods in which a genomic sequence is compared with homologous proteins from the same or different species, (ii) methods in which a genomic sequence is compared with a collection of expressed sequence tags (ESTs) or cDNAs, and (iii) methods, where two or more genomic sequences are compared with each other. Examples of gene finding programs of the first type are `procrustes` [85] and `genewise` [22]. In the next section we discuss two example programs which are representative for the second and third type. In all cases a so-called spliced alignment (Fig. 7.15) of query and subject sequence highlights those fragments which are identical or highly similar. Dedicated programs to compute spliced alignments, such as `sim4` [79] or `est_genome` [185], can accommodate large gaps, which may correspond to putative introns. This is only possible if gap costs remain bounded, even if the gap

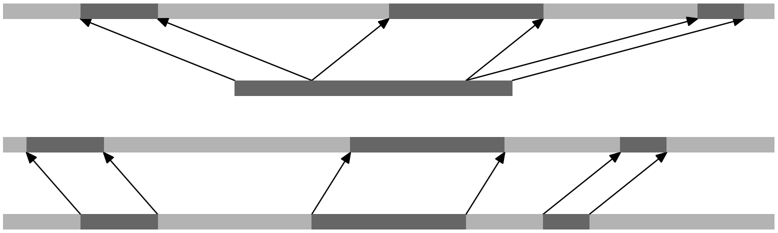


Fig. 7.15. Types of spliced alignments. *Top*: genomic query and protein, cDNA or EST subject sequence; *bottom*: genomic query and homologous genomic subject sequence.

size becomes large. Simple affine linear gap functions, described in Chapter 2, are not suited for this task, but need to be adapted accordingly. At the same time, these programs also attempt to identify potential splice junctions and to insert gaps preferentially at those junctions.

When two (or more) genomic sequences are compared, a strategy to search for exons and splice junctions is to inspect the pattern of sequence similarity in an alignment and the transitions between highly conserved and diverged regions. The underlying idea is that coding sequences are more conserved between species than noncoding sequences. Therefore, the variation in sequence similarity can be used as a quite reliable indicator of the location of protein-coding genes and of the boundaries of (coding) exons and introns. Besides alignments of orthologous sequences (from different species), alignments of paralogous sequences (from the same species) have also been used for this purpose. A disadvantage of these programs is that they have difficulty in locating untranslated exons. Although sequence similarity between orthologous untranslated exons tends to be higher than in introns, it is generally not sufficient to clearly delineate the exact boundaries of the 5’ and 3’ untranslated regions of genes. In principle, untranslated exons and their boundaries can be inferred with the help of ESTs, since they are derived from mRNA. However, the drawback here is that the necessary ESTs may be absent from the EST database.

7.5.2 Comparative Gene Prediction at the *Adh* Locus

As an example for a comparative gene prediction tool, we apply `GeneSeqer` [250] to our test sequence with the two genes *Adh* and *Adh*$_{\mathrm{dup}}$ from *D. melanogaster* (Fig. 7.2). `GeneSeqer` computes spliced alignments of the query and all matching ESTs, and returns putative gene locations, as well as explicit gene structure and protein predictions. A `GeneSeqer` prediction is based on the consensus of a set of clustered, or aggregated, ESTs (Fig. 7.16A). In contrast, the result of a simple BLAST search in the *D. melanogaster*-EST database is shown in Figure 7.16B. Unfortunately, at the time of writing, no *Drosophila* ESTs were available which cover the *Adh*$_{\mathrm{dup}}$ region (position 3226 to 4761). Therefore, any EST-driven gene finder would miss this gene (see Figure 7.16A).

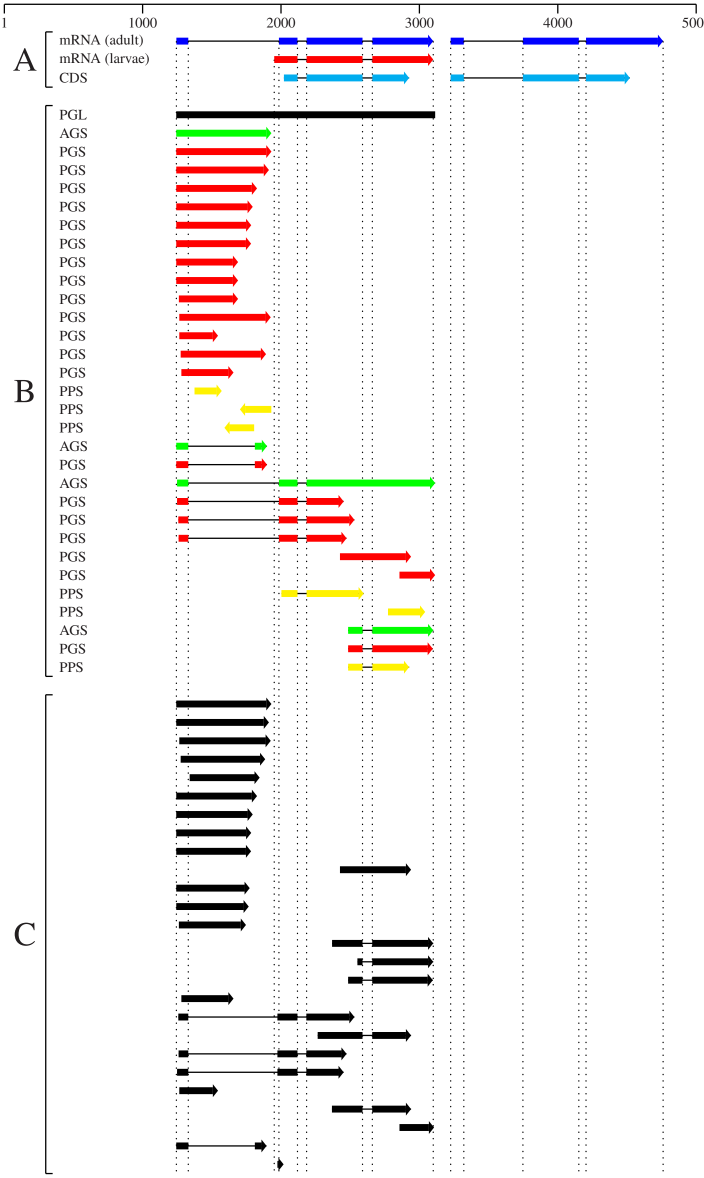

1
1000
2000
3000
4000
5000
A
mRNA (adult)
mRNA (larvae)
CDS
B
PGL
AGS
PGS
PGS
PGS
PGS
PGS
PGS
PGS
PGS
PGS
PGS
PGS
PGS
PGS
PPS
PPS
PPS
AGS
PGS
AGS
PGS
PGS
PGS
PGS
PGS
PPS
PPS
AGS
PGS
PPS
C

Fig. 7.16. Gene prediction at the *Adh* locus of *D. melanogaster*. **A**: Known gene structure (GenBank entries X78384 and M17827). **B**: Graphical output of `GeneSeqer` [250]; PGL—"putative gene location", PPS—"predicted protein sequence", PGS—"predicted gene sequence", AGS—"aggregate gene sequence". **C**: Output of `blastn` when searching the *D. melanogaster* EST database at NCBI (posted date September 27, 2005).

In Figure 7.17 the gene structures predicted by `GeneSeqer` are compared with the ones calculated by `Slam`. The latter is based on a pairwise alignment between two genomic sequences. Here, orthologous sequences of *D. melanogaster* and *D. simulans* were compared. The two fly species diverged about 1-2 million years ago. `Slam` yields a perfect prediction of the coding part of the *D. melanogaster Adh* gene and simultaneously also retrieves the *Adh* gene in *D. simulans*. Furthermore, `Slam` detects part of the *Adh*$_{\mathrm{dup}}$ gene. However, it predicts here a split gene, and only the terminal exon of the *Adh*$_{\mathrm{dup}}$ coding sequence is correctly predicted. Similarity-based gene prediction programs tend to have a high false positive rate when divergence between the two compared sequences is low. This is the case for the two *Drosophila* sequences (see Figure 7.18).

Furthermore, note that `GeneSeqer` is able to correctly identify the 3' untranslated part of the *Adh* gene and nearly correctly the 5' untranslated part, since EST-support is available. In contrast, `Slam` was not able to find untranslated regions. `GeneSeqer` identifies the internal coding exon, but it does not recognize the correct start and stop codons of the *Adh* gene. It even splits its prediction into two different genes (see Figure 7.17). Furthermore, it does not recognize intron 2, but joins exons 2 and 3 into a single exon. *Adh*$_{\mathrm{dup}}$ is completely missed by `GenSeqer`. Finally, note that the EST cluster at the *Adh* locus contains ESTs which support different splice forms. One reason for this is that ESTs usually do not represent the complete transcript (so-called full length cDNA), but only part of it. Sometimes they are contaminated with foreign DNA, for instance vector sequences. A much more important reason, however, is that different ESTs may represent transcript variants and thus be indicative of alternative splicing in this gene. In this case, alternative transcripts are indeed known to exist [152]. Kreitman has described an adult and a larval form of *Adh* (GenBank entry M17827). Nonetheless, none of the currently available ESTs supports the known larval splice variant and it is therefore missed by `GeneSeqer` (see Figure 7.16A and 7.16B).

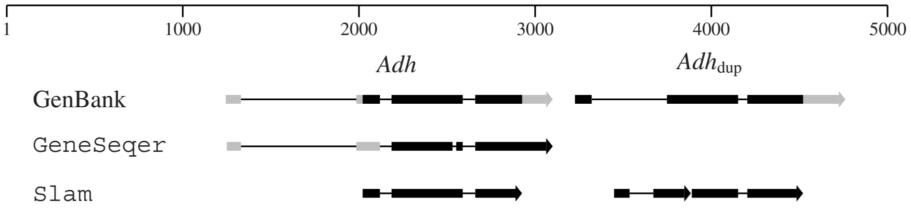


Fig. 7.17. Comparison of GenBank annotation and two comparative gene prediction programs, `GeneSeqer` and `Slam`. The top track shows the GenBank annotation: grey boxes represent the mRNA and black boxes the CDS.

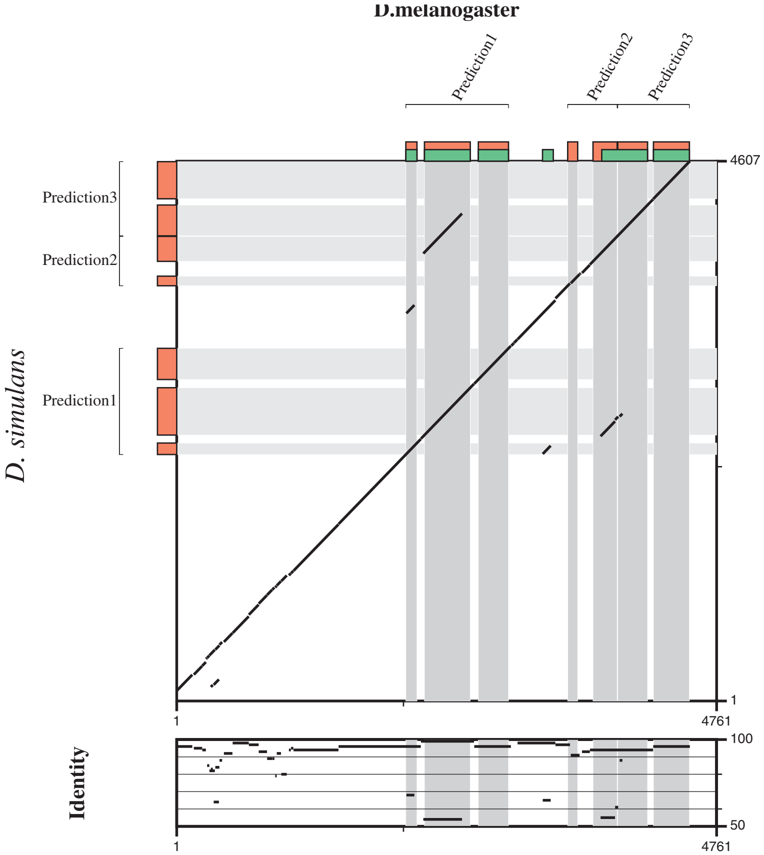


Fig. 7.18. Alignment of two genomic sequences from *D. melanogaster* and *D. simulans*. *Upper panel*: X-axis—*D. melanogaster* sequence (4,761 bp), Y-axis—*D.simulans* sequence (4,607 bp). Lines represent locally aligned segments. Note that the entire region is highly conserved. Dark boxes (along the border) correspond to the `Slam` predicted exons, light boxes represent the annotated exons (only CDS) of the *Adh* and *Adh*$_{\text{dup}}$ genes. *Lower panel*: Identity values of the aligned segments. Note that traces of the gene duplication are still visible: there are short fragments in *Adh* and *Adh*$_{\text{dup}}$ with an identity of about 55%. Since the duplicated fragments occur in both sequences, the duplication event must predate the speciation event leading to *D. simulans* and *D. melanogaster*.

7.6 Problems and Perspectives

Despite the success of automatic gene prediction in whole genome annotation projects (`genome.ucsc.edu`, `www.ensembl.org`), a number of problems remain. Some of them were encountered in the above example. Many programs only predict protein-coding sequences and ignore untranslated regions and regulatory elements of putative genes and non-protein-coding genes altogether. Long introns often lead to erroneous splitting of genes. On the other hand, in regions of high gene density, programs tend to join distinct genes. On a genomic scale it is difficult for gene prediction programs, which are typically trained on a limited set of verified genes, to deal with fluctuating gene densities along a genome. Furthermore, program accuracy is species-specific. This is especially true for all content-based programs, which work with species-specific codon usage tables. Finally, alternative splicing plays an important role in many eucaryotic genes. However, to predict possible transcript variants and to control at the same time the number of combinatorially possible variants turned out to be an extremely difficult problem.

Comparative methods come with their own set of advantages and disadvantages. EST-based gene prediction depends on the availability of a comprehensive EST database. However, it is difficult to assess the degree of completeness of an EST database. Inter-specific as well as intra-specific genomic comparisons require at least two homologous sequences at the appropriate evolutionary distance. Such sequences may not always be available.

Much development effort is spent on combining the advantages of the different gene prediction—or genome annotation—strategies. Developers aim to raise the level of prediction accuracy. However, the exact number of genes in the human or any other complex genome may well turn out to be fundamentally uncertain. Many genes may not have the well-defined “true” structure, which is assumed to exist when evaluating the accuracy of gene predictions. Alternative splicing, differential expression, and interaction with other genes within a complex network make genes much less clear-cut objects than their underlying sequences.

7.7 Summary

Gene prediction is concerned with the automatic detection of genes in nucleotide sequences. Genes consist of a number of features such as exons, introns, promoters, coding sequence, untranslated regions, and transcriptional regulators. These differ widely in the accuracy with which they can be predicted. The accuracy of a given method of gene prediction is quantified through sensitivity and specificity. Sensitivity is the number of correctly detected features divided by the total number of detectable features. Specificity is the number of correctly detected features divided by the number of features returned by a detection method. An ideal gene prediction algorithm is characterized by a sensitivity and a specificity of one. There are two classes of methods for gene finding: *ab initio* and comparative. *Ab initio* methods detect signals contained in the DNA sequence in an attempt to emulate the transcription and translation machinery of a cell. Comparative methods are based on alignments of more than one input sequence. In such alignments functional regions such as exons stand out as islands of elevated sequence conservation.

7.8 Further Reading

Reviews of gene prediction have been published by, among others, Guigó [94], Burge and Karlin [31], and Zhang [264]. An excellent online bibliography on the topic has been compiled by Wentian Li and can be found at `http://www.nslij-genetics.org/gene/`.

7.9 Exercises

7.1. The coordinates of the three exons that make up the CDS of *Adh* in *D. melanogaster* are 2021–2119, 2185–2589, and 2660–2926. What is the reading frame of these three exons?

7.2. Determine a recursive formula, analogous to Equation (7.2), to compute the remainder of exons.

7.3. Determine a formula to convert frame into remainder and *vice versa*.

7.4. Given a uniform distribution of the four nucleotides, what is the number of expected acceptor (donor) dinucleotides in a random sequence of length 100 kb?

7.5. How many stop codons do you expect to find in an exon of length 100 bp in any of the three reading frames on both strands

(a) if the G/C-content is equal to 0.5,
(b) if the G/C-content is equal to 0.3,
(c) if the G/C-content is equal to 0.6?

7.6. What is the average distance between an acceptor and a donor site in a random sequence? Are these distances different in an infinitely long and a finite sequence, say, of length 100 bp (consider here only non-nested exons)?

7.7. Given the log-likelihood matrix in Figure 7.9, what is the expected number of donor sites with positive score ($s > 0$) in a random sequence (all four nucleotides have equal probability) of length 100 kb? Compare this number to the number of expected `GT` dinucleotides.

7.8. Download the set of verified donor sequences listed in Figure 7.10B.

1. Build a frequency matrix and a log-likelihood matrix (PWM) comprising 10 positions (3 upstream + donor dinucleotide + 5 downstream).
2. Determine the information content of each position of this profile.

Part II

Sequences in Time

8

Phylogeny

> *If evolution is true, why are there still monkeys?*
>
> *Larry King [146]*

Biological sequences are the product of evolutionary history and phylogenies are graphical summaries of this history. The simplest and most widely applied model of evolution is a bifurcating tree. As explained previously (Section 3.4), a tree consists of nodes and edges connecting the nodes. The leaves in a phylogenetic tree represent extant sequences of the corresponding organisms. Internal nodes represent common ancestors of all sequences in the respective subtree. In a bifurcating tree each internal node leads to exactly two child nodes. The distances along edges are often interpreted as proportional to time and an example of such a tree is shown in Figure 8.1A, which has been calculated from the globin multiple alignment of Figure 5.1.

As for every phylogenetic tree, there are two important aspects to the globin tree in Figure 8.1A: (i) the topology or branching order of the tree, and (ii) its branch lengths. Consider topology first. The globin tree implies that the α-chain of hemoglobin (α-globin) from human and horse form the most closely related pair of sequences. The next most closely related pair of sequences consists of the β-globins of the same taxa. Together the α- and β-globins form a mammalian globin cluster. The closest relative of this cluster is globin 5 from lamprey, followed by sperm whale myoglobin, and finally leghemoglobin from yellow lupine as the sequence that is the most basal member of the globin protein family.

We can interpret branch lengths in Figure 8.1A as proportional to time. If we knew the mutation rate of globins, we could then calculate the times for each of the internal nodes in the tree. The implicit assumption we are making here is that the mutation rate is equal across the branches. This assumption is known among biologists as the *molecular clock* [28], which is further discussed in Chapter 9. Certain phylogenetic algorithms are based on the assumption of a molecular clock. The vertical line in Figure 8.1A emphasizes that all branches in the corresponding tree end at the same point in time, the present. Other methods do not assume a molecular clock and an example of such a tree for our globin data is shown in Figure 8.1B. Here the interpretation of branch lengths as proportional to time raises the apparent paradox that not all leaves of the tree are aligned along the present. Has globin 5 from lamprey, for example, not yet reached the present? It obviously has and we need to read Figure 8.1B as a summary of the number of mutations estimated to have

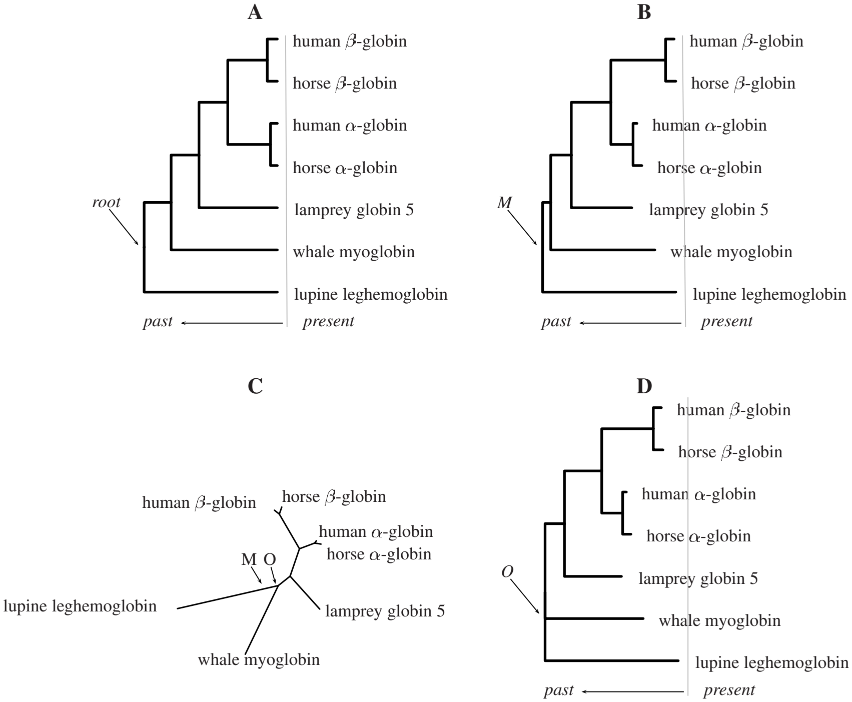


Fig. 8.1. A: Rooted phylogenetic tree of seven globin sequences calculated using the UPGMA algorithm that assumes a molecular clock. The thin vertical line indicates the present point in time. **B**: Midpoint rooted phylogenetic tree of the same globin sequences calculated using the neighbor-joining algorithm that does not assume a molecular clock. **C**: Unrooted version of **B**. **D**: Outgroup rooted version of **B**. M: position of midpoint root; O: position of outgroup root.

occurred between each pair of sequences. Figure 8.1B therefore tells us that in the past the rate of mutation was not identical along all the branches of the topology and in a biological setting one would often go on to test whether these deviations from rate equality were significant [186]. Suffice it to say here that of the algorithms used to reconstruct phylogenies some are based on the assumption of a molecular clock, while others are not. We will make these differences more precise below.

As important as the distinction between algorithms with and without molecular clock is the dichotomy between methods that return rooted and those that return unrooted phylogenies. Formally, this distinction refers simply to the fact that a rooted tree contains a single node (the root) that is not connected to a parent node, while an unrooted tree contains no such node (Fig. 8.2).

In the absence of a molecular clock, the resulting tree is initially unrooted and Figure 8.1C shows the unrooted version of Figure 8.1B. This unrooted tree implies no particular direction for evolutionary time. In other words, Figure 8.1C gives no indication as to the position of the last common ancestor of the clade depicted. There are two widely used methods of determining the position of the root in this tree: midpoint rooting [67] and the more commonly applied rooting by outgroup. When

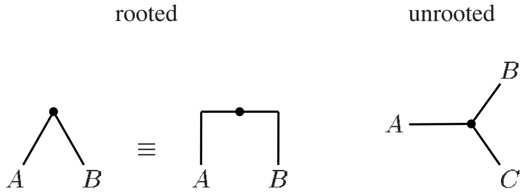


Fig. 8.2. The smallest rooted and unrooted phylogenies.

using midpoint rooting, the root is placed in the middle between the two most divergent taxa, in our case lupine leghemoglobin and horse β-globin. This position is marked as M in the unrooted tree (Fig. 8.1C). When the tree is rooted at this point, we get the topology shown in Figure 8.1B. This tree implies a constant rate of evolution between the two most divergent taxa and accordingly horse β-globin and lupine leghemoglobin are both flush at the imaginary *present* line.

Alternatively, we may note that the most recent common ancestor of leghemoglobin and the other six globins must have been the most recent common ancestor of all our seven example globins. Biologists then speak of leghemoglobin as being the *outgroup* with respect to the remaining six globin sequences. An outgroup can be used to assign a "pseudo root" to a tree. This is done by rearranging the tree around the node O in Figure 8.1C, which is the node at which the outgroup joins the tree. The result of this rearrangement of branches is shown in Figure 8.1D. Notice that node O is structurally the same as all other internal nodes in the tree as it is connected to three other nodes. Recall that in a bifurcating phylogeny a true root is connected to only two other nodes (Fig. 8.2), hence our designation "pseudo root".

We have already seen that we might not be able to model evolution by a constant-rate mutation process, which results in an unrooted tree possibly with uneven branch lengths. Perhaps more worryingly, there might not even be a tree.

In the subsequent sections we start by investigating whether there is a tree to be discovered in the first place. This is followed by a description of three widely used classes of methods for phylogeny reconstruction: (i) distance-based methods, which are also used to construct guide trees in progressive multiple sequence alignment algorithms (cf. Section 5.3), (ii) maximum parsimony, and (iii) maximum likelihood. We finish the chapter explaining how to assess the robustness of the clusters in our phylogenetic tree through a simple but powerful resampling technique known as the bootstrap [56, 71]. Readers interested in more details may wish to consult the comprehensive monograph on phylogenetic reconstruction by Felsenstein [73].

8.1 Is There a Tree?—Statistical Geometry

Statistical geometry is a method for assessing the phylogenetic signal contained in a multiple sequence alignment. It is easiest to explain for binary sequences and we concentrate on this case here. You might wonder what relevance binary sequences might possibly have for biological sequences, given that DNA is an alphabet over four bases, not two. However, the bases can be recoded into purines and pyrimidines yielding binary sequences. Consider, for example, the hypothetical quartet of aligned sequences shown in Figure 8.3A. There are eight possible configurations for each column and an example for each is given in Figure 8.3A: either all four nucleotides are identical, in which case the position is designated as "0" and ignored in the subsequent analysis; or one nucleotide is different from the other three, in which case there are four ways of obtaining this configuration, designated A, B, C, and D; finally, the quartet might divide into two pairs of equal nucleotides, in which case there are three possibilities, designated X, Y, and Z (Fig. 8.3A). Having thus classified each position for a quartet of sequences, the next step in statistical geometry is to count the number of occurrences of each configuration. The result is a three-dimensional diagram, which correctly depicts the mutational distance (of four) between all four sequences (Fig. 8.3B). If more than four sequences are analyzed, all $\binom{n}{4}$ possible quartets that can be formed from n sequences are investigated and averages calculated for the seven parameters of interest, hence the name *statistical* geometry.

A

S_1	`YRYYYYRR`
S_2	`YYRYYYYY`
S_3	`YYYRYRRY`
S_4	`YYYYRRYR`
Class	`0ABCDXYZ`

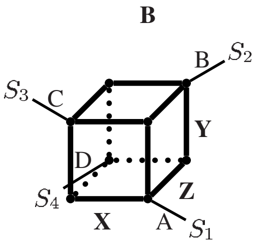


Fig. 8.3. Statistical geometry. **A**: Alignment of four sequences consisting of purines (`R`) and pyrimidines (`Y`) and the class of sequence configuration each column is assigned to. **B**: Depiction of the classes detected in **A**. Box dimensions are marked by bold lines and labels.

The method allows us to distinguish three principal topologies that might describe the data from which phylogenetic reconstruction is to be attempted: the tree, the net, and the bush. A bush is obtained if all three of the box dimensions of the net shown in Figure 8.3 are zero, collapsing the box into a point from which the taxa emerge forming what is known as a "star" phylogeny (Fig. 8.4). If we start from a bush and expand only one of the three box dimensions, we obtain the three possible tree topologies (Fig. 8.4). If we expand more than one box dimension, we obtain the four possible net topologies (Fig. 8.4). Since its inception [60], statistical geometry has been further developed into likelihood-mapping which is explained next.

8.2 Likelihood-Mapping

We start again with the simplest case of phylogenetic reconstruction and consider a quartet of taxa, A, B, C, D. Any given quartet of taxa can be connected by three

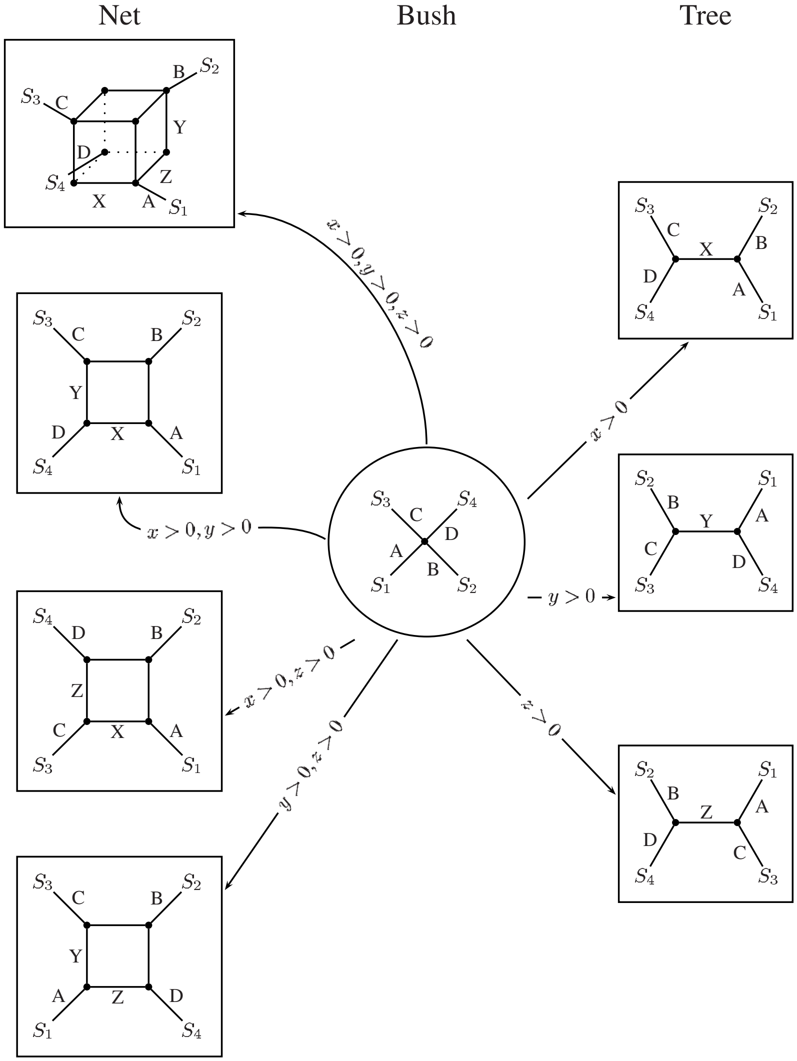

Net
Bush
Tree
$x>0, y>0, z>0$
$x>0, y>0$
$x>0, z>0$
$y>0, z>0$
$x>0$
$y>0$
$z>0$
S_1
S_2
S_3
S_4
A
B
C
D
X
Y
Z

Fig. 8.4. The phylogeny of a quartet of taxa might approximate either a net, a bush, or a tree. These topologies can be obtained by starting with the bush and expanding the box dimensions written along the arrows.

different unrooted trees (Fig. 8.5). The *likelihood* of a tree, T, is the probability of observing the input data, D, given the tree: $P(D|T)$. For a set of n sequences the likelihood attached to each of the three possible trees is determined for each of the possible quartets. We are at this point not concerned with the computation of the likelihoods and regard them simply as scores for each of the three trees. Each likelihood score can be normalized by division by the sum of all three likelihoods, resulting in three numbers ranging between 0 and 1. These can be plotted into a triangular coordinate system, where each vertex corresponds to one of the three trees [237]. Such a plot is shown at the top of Figure 8.6 for the $\binom{7}{4} = 35$ quartets that can be formed from our seven globin sequences. If all quartets supported only one tree each, then all 35 points in the diagram would lie on the vertexes of the triangle. However, as displayed in the bottom right triangle in Figure 8.6, this is only true for $22.9\% + 34.3\% + 31.4\% = 88.6\%$ of the quartets. This indicates that overall there is a strong phylogenetic signal in the data, but some parts of the tree will not be perfectly resolved. Before we proceed to reconstruct a phylogeny, let us consider how many possible topologies there are to choose from.

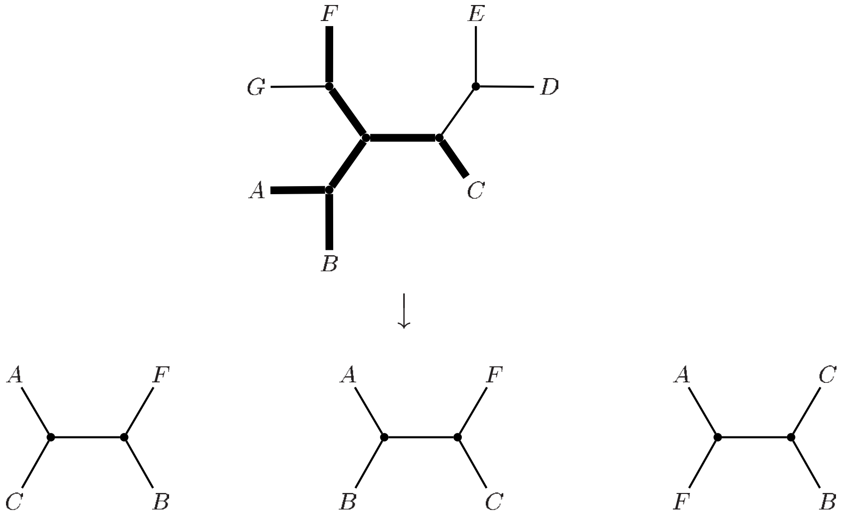


Fig. 8.5. Quartet analysis. *Top*: A tree consisting of the seven taxa A, B, C, D, E, F, G can be divided into $\binom{7}{4} = 35$ quartets of taxa. An example quartet is shown in bold. *Bottom*: the three possible topologies of the example quartet; the tree in the middle corresponds to the quartet marked bold in the top tree.

8.3 The Number of Possible Phylogenies

The simplest unrooted tree consists of three taxa and has a single possible topology (Fig. 8.2). The simplest unrooted tree with more than one topology connects four

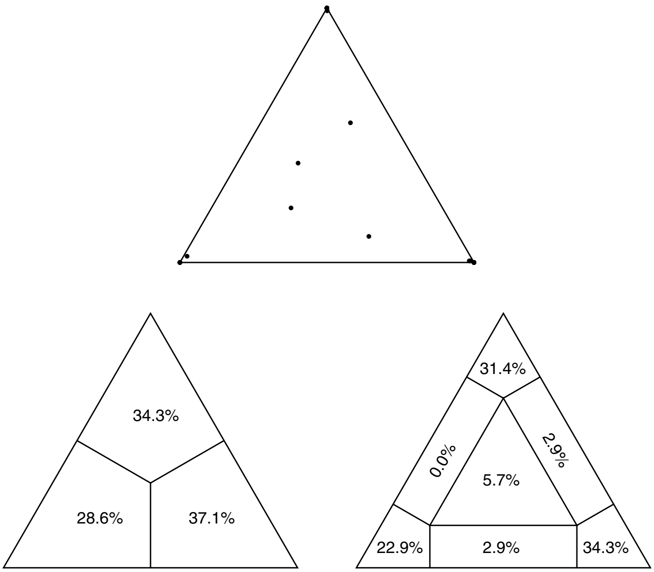


Fig. 8.6. Likelihood-mapping [237] of the phylogenetic content of the globin multiple sequence alignment shown in Figure 5.1. The likelihood-mapping was carried out using the program `TREE-PUZZLE` [218].

taxa and there are three possible configurations (Fig. 8.5). If we add a fifth taxon to our tree, this can be joined to each of the five branches in each of the three trees, i.e. there are $3 \cdot 5 = 15$ possible unrooted trees for five taxa. In general there are

$$N_u = 1 \cdot 3 \cdot 5 \cdot \ldots \cdot (2n-5) = \prod_{i=3}^{n}(2i-5)$$

possible unrooted trees.

The simplest rooted tree consists of two taxa and it has only a single topology (Fig. 8.2). The simplest rooted tree with more than a single possible topology connects three taxa and there are three possible configurations (Fig. 8.7). In order to understand how many rooted trees there are for n taxa, consider the process by which

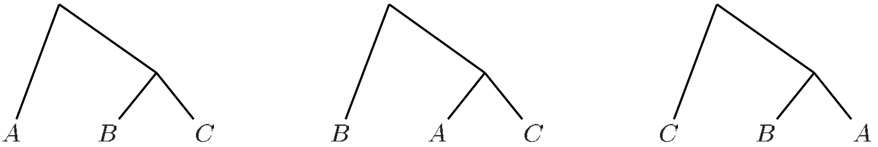


Fig. 8.7. The three possible rooted phylogenies connecting taxa A, B, C.

an unrooted tree is converted into its rooted version. This is done by splitting any of the edges and connecting the free ends of the edges to the root (Fig. 8.8). An unrooted tree contains $2n - 3$ edges. Since any of the edges can receive the root, the number of rooted phylogenies is

$$N_r = 1 \cdot 3 \cdot 5 \cdot \ldots \cdot (2n-3) = \prod_{i=2}^{n}(2i-3).$$

Thus, N_r is a simple function of the number of unrooted trees, N_u:

$$N_r = (2n-3)N_u.$$

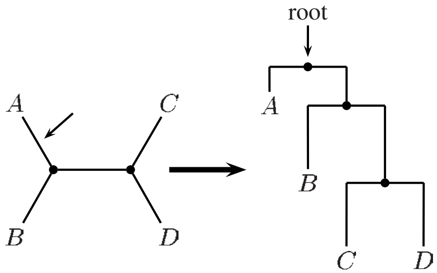


Fig. 8.8. Rooting a phylogeny. The unrooted phylogeny on the left is rooted at the position indicated by the arrow. The resulting rooted phylogeny is shown on the right. The root can be placed on any of the five edges of the unrooted phylogeny.

For merely 20 taxa there are $2.2 \cdot 10^{20}$ unrooted and $8.2 \cdot 10^{21}$ rooted phylogenies. This raises the question of how to choose efficiently between the vast number of alternative phylogenies. There are two general approaches to this problem: (i) to search tree space or (ii) to use an algorithm that avoids this. Searching tree space necessitates a score function for the trees visited, and either the length of the tree in terms of implied mutational steps (maximum parsimony, Section 8.5) or the likelihood of the tree given the data (maximum likelihood, Section 8.6) are used for this purpose [238]. On the other hand, distance methods of phylogenetic reconstruction (Section 8.4) directly calculate a phylogeny based on $\binom{n}{2}$ pairwise distances between n taxa.

8.4 Distance Methods

Before we can apply distance methods, we need to be able to measure the distances between sequences. A naïve method for calculating distances between aligned sequences is to count the pairwise mismatches. However, this underestimates the true number of substitution events per site, as a site might change more than once in the course of evolutionary history. The simplest model to correct for this effect is known as the Jukes-Cantor model [130]. As shown in Section 9.5, the number of substitutions per site over time t, $K(t)$, can be estimated under the Jukes-Cantor model from pairwise sequence alignments using the formula

$$K(t) = -\frac{3}{4} \log \left(1 - \frac{4}{3} P_{\text{diff}}(t) \right),$$

where $P_{\text{diff}}(t)$ is the probability that two homologous nucleotides differ after divergence time t. This quantity is estimated from a multiple alignment by counting the number of pairwise mismatches and dividing by the total number of residues considered. Figure 8.9 shows an alignment of five primate nucleotide sequences. This contains 16 polymorphic positions, all of which have only two nucleotide states, that is, are bi-allelic. The number of pairwise differences and the corresponding Jukes-Cantor distances are displayed in Figure 8.10.

```
Human       GTAAATATAGTTTAACCAAAACATCAGATTGTGAA   35
Chimpanzee  .....t.........c........c..........   35
Gorilla     .....t.........c........c..........   35
Orangutan   .....t.........c........t..........   35
Gibbon      .....c.........t........t..........   35


                 ↓
Human       TCTGACAACAGAGGCTTACGACCCCTTATTTACCG   70
Chimpanzee  ...g.c..c..a.g.tcacg...c...at......   70
Gorilla     ...g.t..c..a.g.tcaca...c...at......   70
Orangutan   ...a.t..t..g.c.ccaca...c...at......   70
Gibbon      ...a.c..t..a.g.tcgaa...t...gc......   70
```

Fig. 8.9. Alignment of 70 nucleotides of mitochondrial DNA sequences from five primates containing 16 polymorphic sites. Dots indicate a match with the nucleotide in the top row; ↓: homoplasy as explained in Section 8.5. Data taken from [111].

Distances that fit onto a tree with a molecular clock (cf. Fig. 8.1A) are known as *ultrametric*. Ultrametric distances obey the *three point criterion*, which states that for any three taxa A, B, C, there are two equidistant pairs of taxa, with the distance between the third being less or equal to the other two [238]. Consider, for example, the topology shown in Figure 8.11, where $d_{AB} = d_{AC} \geq d_{BC}$.

Distances that fit onto a tree without a molecular clock (cf. Figs. 8.1C and 8.1D) are called *additive*. Additive distances obey the *four point criterion* [238]. Consider four taxa, A, B, C, D. Between these four taxa six distances and three pairs of distances can be formed. The four point criterion states that the sum of those pairs of distances that traverse the trunk of the tree are equal and greater, or equal to the sum of the remaining distances. The topology shown in Figure 8.12 gives an example of this. Ultrametric distances are additive, but not *vice versa*. The UPGMA tree shown

	Human	Chimpanzee	Gorilla	Orangutan	Gibbon
Human	-	0.014	0.044	0.141	0.195
Chimpanzee	0.014	-	0.029	0.124	0.176
Gorilla	0.043	0.029	-	0.091	0.176
Orangutan	0.129	0.114	0.086	-	0.176
Gibbon	0.171	0.157	0.157	0.157	-

Fig. 8.10. Distance data computed from the alignment shown in Figure 8.9. The bottom triangle of the matrix contains the number of mismatches per position, the top triangle displays the corresponding Jukes-Cantor distances, that is, the estimated number of substitutions per position. The Jukes-Cantor distance is always greater or equal to the number of mismatches per position, as it corrects for the possibility of a mismatch corresponding to more than a single substitution event.

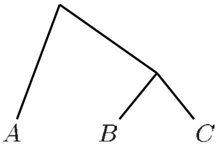


Fig. 8.11. Example of ultrametric distances, where $d_{AB} = d_{AC} \geq d_{BC}$.

in Figure 8.1A corresponds exactly to the distances in the input matrix if these are ultrametric. In contrast, the neighbor-joining tree in Figure 8.1B exactly reflects the distances in the input data if these are additive.

The UPGMA algorithm is an example of average linkage clustering. This is explained next, followed by the neighbor-joining algorithm.

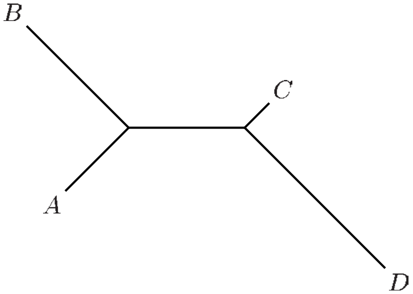


Fig. 8.12. Example of additive distances, where $d_{AD} + d_{BC} = d_{BD} + d_{AC} \geq d_{AB} + d_{CD}$.

8.4.1 Average Linkage Clustering

Average linkage clustering refers to a group of distance algorithms. The most widely used member of this group is the **u**nweighted **p**air-**g**roup **m**ethod using an **a**rithmetic average (UPGMA) [232]. If our distance data is ultrametric, UPGMA will recover the correct tree. We start with only the leaves of our tree. The first internal node is found from the distance matrix by grouping those two taxa that have the smallest distance between them. These taxa are removed from the data set and replaced by cluster c. The distance between c and each of the subsumed leaves is just half the distance between the leaves. The distance between c and some other cluster currently contained in the data set, k, is the arithmetic average of the distances between members of c and k. Hence, the algorithm performs the following steps:

1. Input is a matrix of pairwise distances $D = (d_{ij})$ between a set of taxa $\mathcal{T} = \{1, 2, \ldots, n\}$
2. Find pair of taxa $\{i, j\}$ with smallest distance
3. Cluster taxa: $c \leftarrow \{i, j\}$
   Remove pair $\{i, j\}$ from set of taxa: $\mathcal{T} \leftarrow \mathcal{T} - \{i, j\}$
   Add new cluster to set of taxa: $\mathcal{T} \leftarrow \mathcal{T} \cup \{c\}$
   if $|\mathcal{T}| = 2$, stop
4. Compute the distances between c and all other taxa:

$$d_{ck} = \frac{d_{ik} + d_{jk}}{2}, \; k \in \mathcal{T};$$

   if the input data is ultrametric, this is always the same number
5. go to 2

	1	2	3	4
1	-			
2	6	-		
3	2	6	-	
4	6	4	6	-

	5	2	4
5	-		
2	6	-	
4	6	4	-

	5	6
5	-	
6	6	-

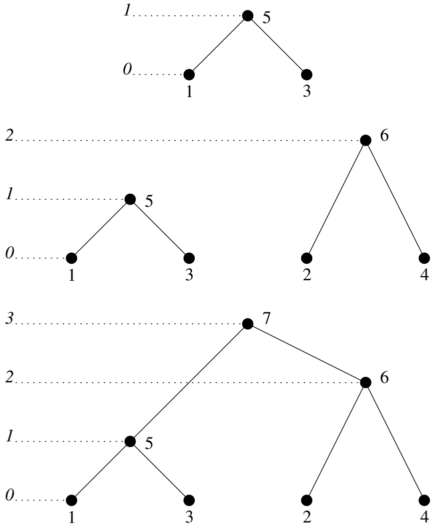


Fig. 8.13. Phylogenetic reconstruction using the UPGMA algorithm. As the distance matrix shrinks, the phylogenetic tree grows.

Figure 8.13 illustrates these steps. Notice that ultrametricity of the distance data implies that the number of distinct entries in the distance matrix must not exceed the number of internal nodes (including the root) in the tree. The number of internal nodes in a rooted binary tree for n taxa is $n-1$ and, hence, if a given distance matrix contains more than $n-1$ distinct entries we know that the distances are not ultrametric. The distance matrix shown in Figure 8.10 is an example of this. It contains 8 distinct entries, which is greater than $5-1=4$. We can still apply the UPGMA algorithm to this data and the result is shown in Figure 8.14. This tree agrees with the generally accepted primate phylogeny. However, in other cases a violation of the assumption of ultrametricity can lead to an incorrect phylogeny. Consider for example the phylogeny shown in Figure 8.15. The rates of evolution along the branches leading to taxa **2** and **3** are much lower than the rates along the branches leading to taxa **1** and **4**. UPGMA would first cluster taxa 2 and 3 thus returning the wrong tree. In contrast, the neighbor-joining method uncovers the correct tree from the distance matrix shown in Figure 8.15.

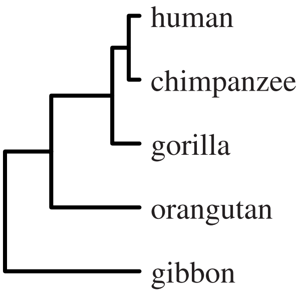
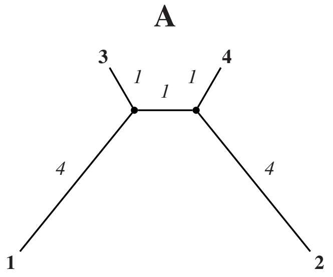


Fig. 8.14. Primate phylogeny computed from the data shown in Figure 8.10 using the UPGMA algorithm.



B

	1	**2**	**3**	**4**
1	0	5	6	9
2		0	3	6
3			0	5
4				0

Fig. 8.15. **A**: Phylogeny with varying rates of evolution; **B**: corresponding distance matrix. Branch lengths are italicized. The neighbor-joining method can recover tree **A** from data **B**, while the UPGMA algorithm would erroneously cluster taxa **2** and **3**.

8.4.2 Neighbor-Joining

Pairs of nodes in a phylogenetic tree that are connected through a single third node are called *neighbors*. For example, in the final tree shown in Figure 8.13 nodes 2 and 4 are neighbors and so are nodes 5 and 6, but not nodes 2 and 3. The neighbor-joining method proceeds by calculating for each possible pair of neighbors the total length of the branches of the corresponding tree. At each stage the pair of neighbors is chosen which induces the shortest tree. This method recovers the correct phylogeny from additive distance data [212] in the same number of steps as the UPGMA method explained above. As with the UPGMA algorithm, we start with an n by n distance matrix. The neighbor-joining algorithm then consists of four iterated steps:

1. For each pair of taxa i, j compute

$$L_{ij} = d_{ij} - \frac{r_i + r_j}{n-2},$$

   where r_i is the sum of the distances from taxon i to all other taxa k:

$$r_i = \sum_k d_{ik}.$$

2. Pick the pair of taxa with the smallest value of L_{ij} and cluster them to form taxon c. The distances from c to the other members of the data set are

$$d_{kc} = \frac{1}{2}\left(d_{ik} + d_{jk} - d_{ij}\right), k \neq i, j.$$

3. Finally, compute the branch lengths from c to taxa i and j:

$$d_{ic} = \frac{1}{2(n-2)}\left((n-2)d_{ij} + r_i - r_j\right)$$

   and

$$d_{jc} = \frac{1}{2(n-2)}\left((n-2)d_{ij} + r_j - r_i\right).$$

4. Replace taxa i and j by c and reduce n by one. If $n > 2$ go to 1. Otherwise, generate the last branch with length $L_{ij} = d_{ij}$ and stop.

To see this algorithm in action, consider again the distance matrix that defeated the UPGMA algorithm in Figure 8.15. Figure 8.16 shows how the neighbor-joining method recovers the correct tree.

Application of the neighbor-joining algorithm to the primate data (Fig. 8.10) results in the phylogeny shown in Figure 8.17. When comparing this to the corresponding UPGMA tree in Figure 8.14 the two main differences, unrootedness and absence of a molecular clock, can clearly be seen.

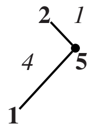
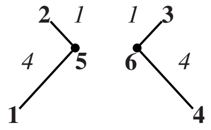
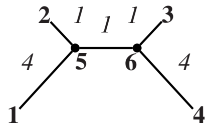
<table>
<tr><th>Step</th><th>Distance Matrix</th><th>Branch lengths</th><th>Tree</th></tr>
<tr><td>1</td><td>$$\begin{array}{c|cccc|c} & 1 & 2 & 3 & 4 & r_i \\ \hline 1 & 0 & 5 & 6 & 9 & 20 \\ 2 & -12 & 0 & 3 & 6 & 14 \\ 3 & -11 & -11 & 0 & 5 & 14 \\ 4 & -11 & -11 & -12 & 0 & 20 \end{array}$$</td><td>$d_{15} = \frac{1}{4}(2 \cdot 5 + 20 - 14) = 4$
$d_{25} = \frac{1}{4}(2 \cdot 5 - 20 + 14) = 1$</td><td> </td></tr>
<tr><td>2</td><td>$$\begin{array}{c|ccc|c} & 3 & 4 & 5 & r_i \\ \hline 3 & 0 & 5 & 2 & 7 \\ 4 & -12 & 0 & 5 & 10 \\ 5 & -12 & -12 & 0 & 7 \end{array}$$</td><td>$d_{36} = \frac{1}{2}(5 + 7 - 10) = 1$
$d_{46} = \frac{1}{2}(5 - 7 + 10) = 4$</td><td> </td></tr>
<tr><td>3</td><td>$$\begin{array}{c|cc} & 5 & 6 \\ \hline 5 & 0 & 1 \\ 6 & & 0 \end{array}$$</td><td></td><td> </td></tr>
</table>

Fig. 8.16. Neighbor-joining method. The top triangle of the distance matrices contains the raw distances, while the corresponding L_{ij}-values are displayed in the lower triangle; hence in step 1 for example, $r_2 = 5 + 3 + 6 = 14$, where the first element of the sum needs to be transposed from the top triangle of the matrix (see text for further details). Nodes are labeled by bold numbers. Italicized numbers indicate branch lengths.

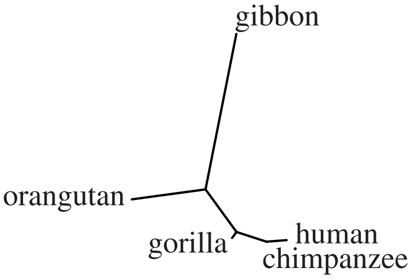


Fig. 8.17. Primate phylogeny computed from the data shown in Figure 8.10 using the neighbor-joining algorithm. The branch leading to chimpanzee is imperceptibly short.

8.5 Maximum Parsimony

In contrast to the distance methods already discussed, both maximum parsimony as well as maximum likelihood (Section 8.6) are strictly speaking only methods for evaluating a given phylogenetic tree. In practice, they are always combined with a search algorithm looking for the best tree. Under maximum parsimony we seek the tree that implies the fewest mutations along its branches. One of the most popular algorithms for calculating the number of mutations implied by a tree has become known as Fitch-parsimony in honor of its inventor Walter Fitch [77]. The algorithm takes as input a set of character data such as a multiple sequence alignment and a tree (Fig. 8.18). The algorithm implies a model of symmetrical mutation and can therefore only be used to derive unrooted trees. However, the tree traversal underlying the algorithm requires a rooted tree as input. This problem is overcome by arbitrarily rooting the given tree. Next, the algorithm goes through every position in the alignment in order to compute the "length" of the tree, $L(T)$. The quantity $L(T)$ can be thought of as the number of mutations implied by the tree. Our input data is a multiple sequence alignment from which an initial rooted tree is constructed. At each alignment position, P_j, the following steps are carried out:

	P_1	P_2	P_3
W	A	A	T
X	C	G	C
Y	C	G	C
Z	T	G	T

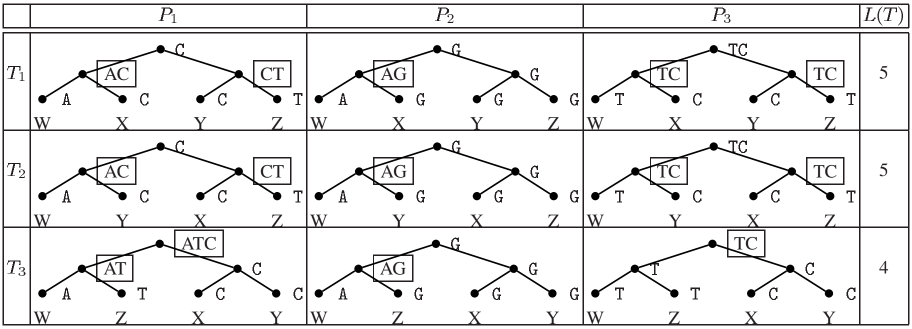


Fig. 8.18. Fitch-parsimony algorithm [77]. The three possible unrooted trees for four taxa T_1, T_2, T_3 get arbitrarily rooted and are then evaluated at each position P_1, P_2, P_3 of the alignment. Each internal node is labeled according to the following convention: if the intersection of the labels on the child nodes is empty, the union of the child labels is taken, otherwise the intersection. Labels generated through the union operation are boxed. The number of boxed labels is summed to generate the "length" $L(T)$ of a tree.

1. Label the leaves with the character of the corresponding taxon at P_j.

2. Carry out a depth-first traversal of the tree and label each internal node by forming the intersection of the labels of its two child nodes. If this intersection is empty, the parent node is labeled by the union of the child labels, otherwise it is labeled by the intersection. In case of an empty intersection, $L(T)$ is increased by one.

Once all trees have been evaluated, the shortest tree is chosen as the most parsimonious evolutionary hypothesis. This procedure is illustrated in Figure 8.18. Here the shortest tree is T_3 and, hence, under the maximum parsimony criterion it is selected as the best tree. Notice, however, that trees T_1 and T_2 have the same value of $L(T)$. In fact, the parsimony length of a tree can only vary between the number of variable positions in the alignment, S, and half the number of its branches times S. The upper limit is due to the fact that at each variable position in the alignment a maximum of one mutation per pair of child nodes rooted on the same parent node can be accumulated. A rooted tree has $2(n-1)$ branches, where n is the number of taxa considered, and, hence,

$$S \leq L(T) \leq S(n-1).$$

The value of $S(n-1)$ is usually much smaller than the number of possible trees and as a consequence, numerous tree topologies may correspond to a given tree length.

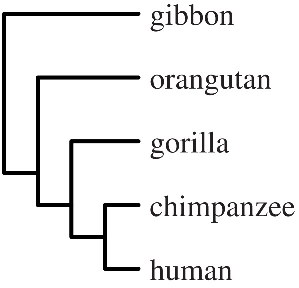


Fig. 8.19. Maximum parsimony phylogeny computed from the data shown in Figure 8.9. Rooting by designating gibbon as the outgroup.

For our primate example data in Figure 8.9, $S = 16$. The most parsimonious tree computed from this data is shown in Figure 8.19. Notice that this topology contains no information about branch lengths, only about branching order. It is unique and has length 17, that is one step more than the theoretical minimum necessary to account for the data. In other words, the alignment must contain one position that can only have arisen from two mutations. This position is marked by an arrow in Figure 8.9 and implies that, given the tree in Figure 8.19, either the `T` or the `C` must have arisen twice in the course of evolution. The reasoning underlying this conclusion is illustrated in Figure 8.20. A character that arises twice in the course of evolution is known as a *homoplasy* and the number of homoplasies on a maximum parsimony phylogeny is $L(T) - S$.

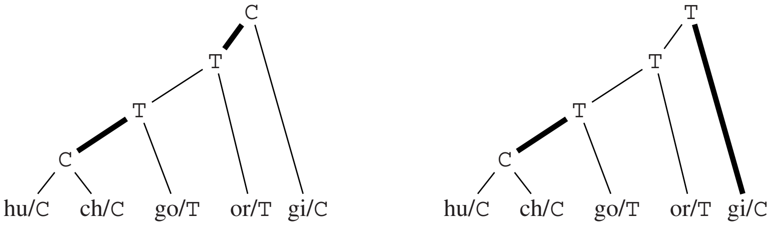


Fig. 8.20. Homoplasy in position 41 of the primate sequence data in Figure 8.9. The two alternative reconstructions of ancestral character states each imply two mutations, shown as bold branches.

It was said that the tree length $L(T)$ equals to a first approximation the number of mutations implied by the phylogeny. Now, the number of mutations that might have taken place in the history of a sequence sample varies between zero and infinity when in fact we have seen that under the parsimony criterion $L(T)$ is bounded above by $S(n-1)$. Equating $L(T)$ with mutations is therefore only reasonable under a model of evolution where each character analyzed has mutated only once. This is approximated in reality by closely related DNA sequences, such as those shown in Figure 8.9. For highly divergent sequences the parsimony criterion may lead to an increasingly rapid convergence on an incorrect tree as more and more data are analyzed [69]. This problem is overcome by using an explicit model of evolution and searching for the tree which has the greatest likelihood of having generated the observed data under the model. This method is known as "maximum likelihood".

8.6 Maximum Likelihood

Maximum likelihood is a framework for statistical inference. Under this framework the central quantity considered is the probability, P, of observing some data, D, given a hypothesis, H, about how this data was generated: $P(D|H)$. This probability is called the *likelihood* of H. In maximum likelihood inference we seek the hypothesis with the highest likelihood.

Consider for example the DNA sequence `TACTTCCTGT`. We wish to infer its GC-content, p. If we assume that positions in our sequence evolve independently from each other and that the GC-content is uniform across the sequence, we can calculate the probability of the observed sequence data given p, $P(D|p)$, by noting that it contains 4 `G`/`C` and 6 `A`/`T` residues, and remembering that the probability of observing an `A` or a `T` is the complement of the probability of observing a `G` or a `C`:

$$L = P(D|p) = p^4(1-p)^6. \tag{8.1}$$

The graph of this likelihood function is displayed in Figure 8.21.

We now wish to compute the maximum of Equation (8.1) by taking its derivative with respect to p, setting the result equal to zero and solving for p. This is best done after first taking the logarithm of Equation (8.1):

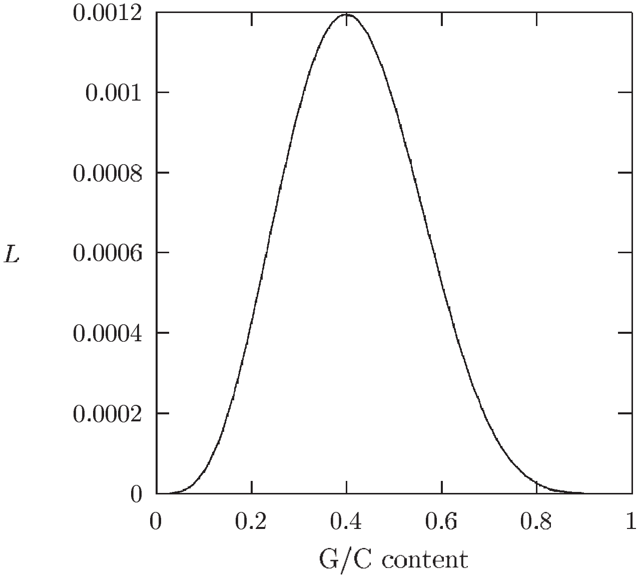


Fig. 8.21. The likelihood of the DNA sequence `TACTTCCTGT` as a function of its GC-content, p. This is a plot of Equation (8.1).

$$\ln L = 4 \ln p + 6 \ln(1-p)$$

and differentiating

$$\frac{d(\ln L)}{dp} = \frac{4}{p} - \frac{6}{1-p} = 0,$$

which yields $\hat{p} = 2/5$. This is equal to the fraction of `G`/`C` residues in the original sequence. In fact, the fraction of `G`/`C` residues is the maximum likelihood estimator of p.

Maximum likelihood was introduced into statistics by the English population geneticist Fisher and first applied to phylogenetic reconstruction by Edwards and Cavalli-Sforza in 1964 [55]. Given some data, D, and an evolutionary model such as the Jukes-Cantor model, M, the most likely tree is the tree that maximizes the probability of the data, given the model and the tree: $\max_T(P(D|T,M))$. The search for $\max_T(P(D|T,M))$ is computationally expensive as it ranges over all possible states at each internal node in a given tree, over all possible branch lengths in that tree and over all variable positions in the underlying multiple alignment. An efficient algorithm for estimating these branch lengths based on dynamic programming was introduced by Felsenstein in 1981 [70].

Figure 8.22 displays the maximum likelihood phylogeny based on the primate data of Figure 8.9. The tree has a log-likelihood of -166.52. If we switch the position of chimpanzee and gorilla in Figure 8.22 the resulting topology has a log-likelihood of -169.23. In order to test whether the two topologies are significantly different,

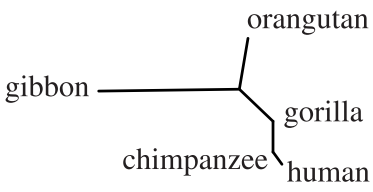


Fig. 8.22. Maximum likelihood phylogeny computed from the primate sequence data shown in Figure 8.9. Notice that the branches leading to chimpanzee and gorilla are imperceptibly short.

the likelihood ratio test can be applied [70, 125]. This is a general statistical test of the goodness-of-fit between two alternative models. The test is based on the ratio, R, of the likelihoods of the two models being compared:

$$R = 2\ln\left(\frac{L_1}{L_2}\right) = 2(\ln L_1 - \ln L_2).$$

R approximately follows a χ^2-distribution. For our example, $R = 2(169.23 - 166.52) = 5.42$. Since the phylogeny has $2n - 3 = 7$ branches whose lengths were estimated with a one-parameter substitution model (Jukes-Cantor model), we have seven degrees of freedom. For $\alpha = 0.05$ and seven degrees of freedom the critical value of the χ^2-distribution is 14.07 and we conclude that based on the data in hand we cannot reject the alternative tree where gorilla is the closest relative of man instead of chimpanzee.

8.7 Searching Through Tree Space

Optimality criteria such as parsimony or likelihood still leave the question open of how to efficiently search the vast space of possible phylogenies. There is no solution to this problem available that runs in polynomial time and is guaranteed to find the optimal tree. Therefore, a lot of thought has gone into designing heuristic searches in “tree space” as well as shortcuts in the full search.

The central problem of heuristic search methods is that they might lead to a local instead of the global optimum. The probability of ending up with a suboptimal search result depends not only on the method of navigating tree space, but also on the strategy of pursuing a better tree once a candidate for optimality has been found. A “greedy” approach would take the first tree with, say, a better likelihood and investigate its neighborhood of trees. However, such a strategy might quickly lead to a local optimum with little opportunity for finding a better tree that is separated from the present tree by one or more intermediate trees with a lower score (Fig. 8.23).

In the following sections we start by looking at two heuristic search methods, nearest neighbor interchange and subtree pruning and regrafting, before explaining an efficient optimal method for searching tree space known as “branch and bound”.

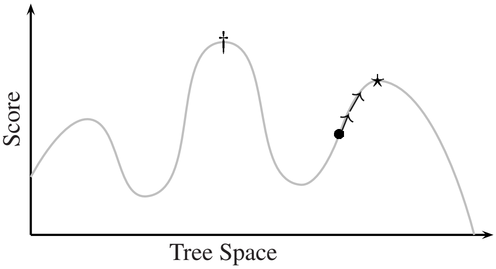


Fig. 8.23. Searching in one-dimensional tree space. Each tree has a score attached, for example, a likelihood. The dot indicates the starting tree. A greedy search strategy that always pursues the next tree that is better than the present one would lead to the local optimum (★) instead of the global optimum (†).

8.7.1 Nearest Neighbor Interchange

An intuitive method for moving in tree space is to consider a quartet of subtrees connected by an internal branch and to look at the two alternative topologies this quartet may have. In an unrooted bifurcating phylogeny there are $n - 3$ internal branches, which means that $2(n - 3)$ trees can be reached from any given tree using nearest neighbor interchange (Fig. 8.24). This is in most cases a very small proportion of tree space and we may think of nearest neighbor interchange as a method for examining the "microenvironment" of a given tree in tree space. However, any one of the trees

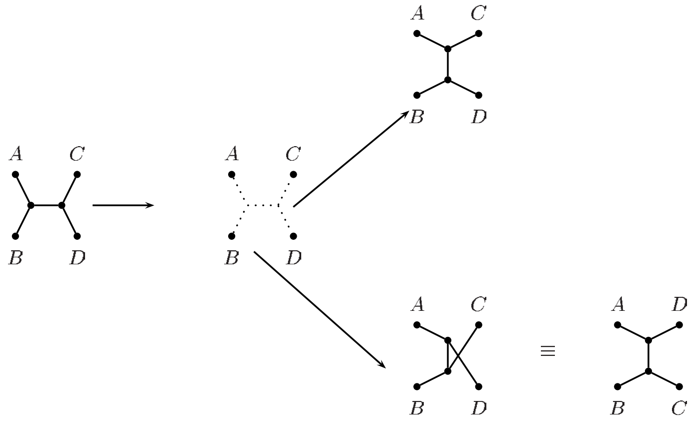


Fig. 8.24. Nearest neighbor interchange. A quartet of subtrees is rearranged into the three alternative possible topologies.

reached by nearest neighbor interchange may become the starting point for a new round of nearest neighbor interchange, resulting in a higher portion of the tree space searched.

8.7.2 Subtree Pruning and Regrafting

An alternative method for searching tree space is illustrated in Figure 8.25: a subtree is removed and reinserted at a new position. If in a tree for $n = n_1 + n_2$ taxa we remove a subtree containing n_2 species, we obtain $2n_1 - 3$ possible points to reinsert the subtree. One of these is the original point and, hence, we get $(2n_1 - 3 - 1) + (2n_2 - 3 - 1) = 2n - 8$ neighbors at each internal branch. For each external branch $2n - 6$ neighbors can be inferred. Since there are n exterior and $n - 3$ interior branches, the total number of neighbors that can be reached is $n(2n - 6) + (n - 3)(2n - 8) = 4(n - 3)(n - 2)$. Not all of these are distinct but nevertheless, subtree pruning and regrafting leads to a wider search of tree space than nearest neighbor interchange.

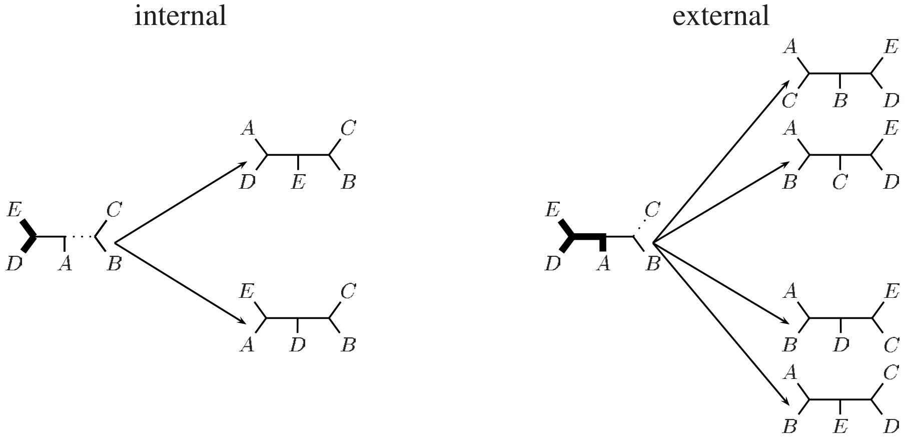


Fig. 8.25. Pruning and regrafting of internal and external branches on a tree with five taxa. Dotted lines indicate pruned branches and bold lines indicate branches receiving a graft.

8.7.3 Branch and Bound

The space of all possible trees describing a given number of taxa can itself be depicted as a tree (Fig. 8.26). The method of branch and bound starts from the smallest possible tree consisting of three taxa. Going outward from the original tree, successive topologies are generated by sequentially adding a new branch to each internal branch of the ancestral tree. Each tree generated along this outward path is evaluated by, say, the parsimony criterion. Once the first set of trees consisting of all taxa have been found, their smallest parsimony score becomes the upper limit for any as yet unfinished topologies awaiting completion. If the best complete topology found so far has a parsimony score of s, then any subtree in the tree of trees rooted on a topology that has a score of $\geq s$ is ignored. The example in Figure 8.26 demonstrates this principle. The best tree found at the depicted stage of the algorithm has length 7. Hence the trees that can be reached from internal nodes (of the tree of trees) whose score is 7 need not be considered, as their lengths must be > 7. This strategy potentially saves a lot of computation time while still guaranteeing that the best tree is found.

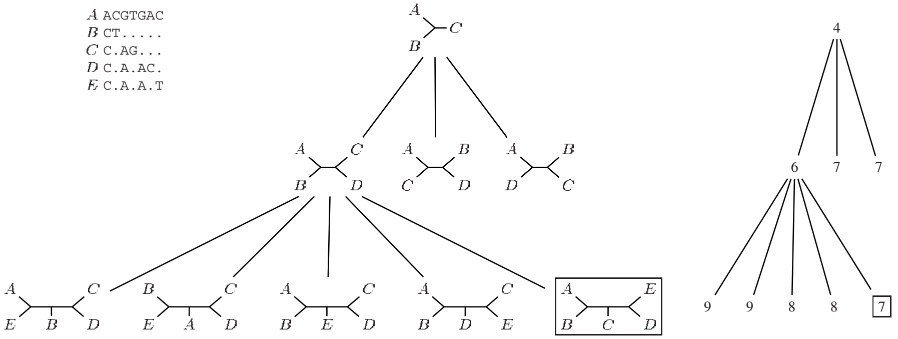


Fig. 8.26. Branch and bound traversal of tree space. *Left*: Alignment of DNA sequences taken from five taxa, $A \ldots E$; dots indicate identity to nucleotide in top row. *Middle*: The nine out of a total of 19 topologies that need to be assessed during a search for the most parsimonious unrooted phylogeny of $A \ldots E$. *Right*: Number of mutations implied by the corresponding topologies. The optimal phylogeny and its score are boxed.

8.8 Bootstrapping Phylogenies

In Section 8.2 we used likelihood-mapping to show that the phylogenetic signal in the globin alignment was quite strong, though not perfect. This indicates that some parts of the tree might not be fully resolved, i.e. some of the quartets fit more than one of the three possible trees. There are a number of methods for locating these unresolved regions and the bootstrap is perhaps the most widely used approach to this problem.

Classical statistical parameter estimation is usually based on the following scenario: Given a sample and the estimator of a parameter calculated from this sample, how would this estimator fluctuate if more samples were collected and the estimator was recalculated from each of the new samples? In practice it is often not feasible to obtain more samples. However, if the distribution of the desired statistic is known, a confidence interval for the initial parameter estimation can be computed. Unfortunately, in many cases this so-called null distribution of the estimator in question is unknown. In such a situation the bootstrap can often be applied successfully. This consists of resampling from the original sample with replacement and recalculating the desired statistic from each of these bootstrap samples to obtain its distribution. The bootstrap was first proposed by Efron in 1979 [56, 57] and first applied to phylogenies by Felsenstein six years later [71].

With phylogenies the bootstrap works as follows. The input data is a multiple alignment of the kind shown in Figure 8.9. This is essentially an m by n matrix of amino acids or nucleotides, where m is the number of sequences and n the length of the alignment.

1. Compute a phylogenetic tree, T, from the multiple sequence alignment and save it.
2. Generate a bootstrap sample by randomly drawing n times with replacement columns of m nucleotides from the original alignment.
3. Compute a tree, T', from the bootstrap alignment using the same algorithm as in 1.
4. Record the clusters in T that also appear in T'.
5. Repeat steps 2–4 many times and sum the number of times a cluster in T also appears in T'.

An example of bootstrapping a multiple sequence alignment is shown in Figure 8.27. Notice two things: (i) some columns of the original sample appear more than once in the bootstrap sample, others are left out. This is due to “sampling with replacement”. (ii) The order in which the columns of nucleotides appear in the bootstrap sample is quite different from their original order. The order of nucleotide columns has no influence on the topology or the branch lengths of a phylogeny. This corresponds to the standard assumption in phylogenetic reconstruction that residues evolve independently.

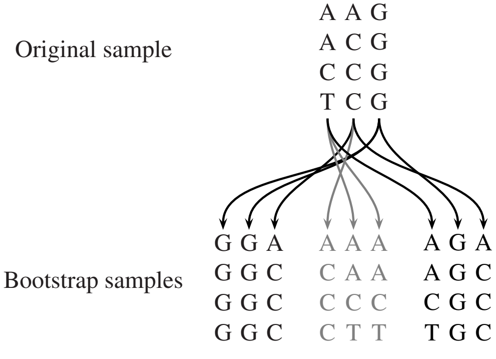


Fig. 8.27. The bootstrap procedure in phylogenetic reconstruction. Each of the bootstrap samples would subsequently be subjected to phylogenetic analysis as further explained in the text.

Figure 8.28 shows the result of applying the bootstrap procedure to our globin alignment. Next to each node there is a number which indicates in how many of the bootstrap trees the same set of leaves of the respective subtree was clustered. The higher this number, the more reliable is the grouping of the corresponding sequences. Notice that the (pseudo) root does not have a bootstrap number. This is because its bootstrap support would always and trivially be maximal.

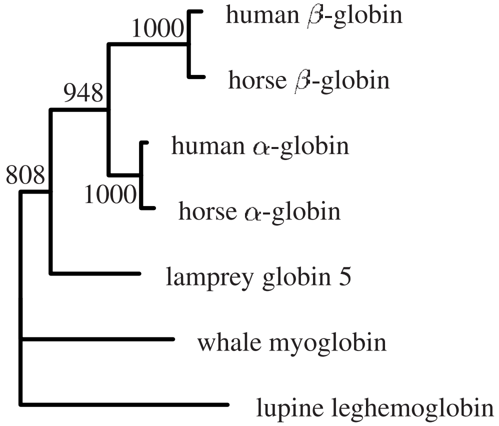


Fig. 8.28. Bootstrapped phylogenetic tree of seven globin sequences using the neighbor-joining algorithm. The numbers next to the nodes indicate how many times the corresponding cluster appeared in 1,000 trees computed from data sets generated from the alignment shown in Figure 5.1 using the bootstrap procedure (cf. Fig. 8.27).

8.9 Summary

Phylogenetic reconstruction is concerned with uncovering the evolutionary history of a group of taxa, given data on homologous characters. The result is usually a bifurcating tree. However, it is *a priori* not clear that the data support a tree. The type of phylogenetic signal contained in the data can be explored using statistical geometry or its more modern version, likelihood-mapping. Phylogenies are either rooted or unrooted. Rooting can be achieved by assigning an outgroup or by assuming a molecular clock. There are three classical approaches to phylogenetic reconstruction: distance methods, maximum parsimony, and maximum likelihood. Distance methods may be based on ultrametric or additive distances. Ultrametric distances conform to a molecular clock and are the assumed input to the UPGMA algorithm, while the neighbor-joining algorithm builds a tree that reflects additive distances. Maximum parsimony and maximum likelihood are methods for *evaluating* a given tree compared to a set of possible alternative trees. The tree length returned by maximum parsimony can be used to quantify the amount of parallel evolution, or homoplasy, implied by the tree. Maximum likelihood tree reconstruction is based on an explicit evolutionary model and allows pairwise comparison of alternative phylogenies by the likelihood ratio test. This gives an overall assessment of the quality of a phylogeny. The bootstrap procedure, in contrast, can be used to assess the robustness of individual subclusters in the tree.

8.10 Further Reading

Swofford and colleagues have written a comprehensive review of phylogenetic reconstruction methods [238]. The monograph by Page and Holmes is a good introduction for those new to the field [194], while Felsenstein provides a more comprehensive and advanced treatment [73].

8.11 Exercises and Study Questions

8.1. Consider the following multiple sequence alignment:

```
A   TCGGTAGGCT
B   ACCGTTCCAT
C   ACCCAAGGCT
D   ATGGTAGGCT
```

How many rooted and unrooted phylogenies can you construct out of the taxa shown in the alignment?

8.2. Compute the pairwise number of mismatches between the taxa in Problem 8.1 and write down the corresponding distance matrix.

8.3. Use the UPGMA algorithm to compute the phylogeny of the taxa in Problem 8.1.

1. Is this phylogeny rooted?
2. Are the distances in the distance matrix ultrametric?

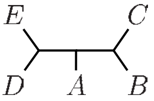
8.4. In Section 8.4.1 it was stated that a necessary condition for ultrametricity is that distances only take $\leq n-1$ distinct values, where n is the number of taxa analyzed. Is this condition also sufficient for ultrametricity?

8.5. Use the `bioinformer` program `Evolution` → `Phylogeny` (Section A.4.1) to compute pairwise distances from the alignment of primate DNA sequences displayed there.

8.6. Use the `bioinformer` program `Evolution` → `Phylogeny` to compute the UPGMA phylogeny from pairwise distance data.

8.7. Use maximum parsimony to find the best tree describing the evolution of the data set shown in Problem 8.1 .

8.8. Consider the following phylogeny



and write down all phylogenies that can be reached using

1. nearest neighbor interchange,
2. subtree pruning and regrafting.

What fraction of the tree space is covered by each method?

8.9. Use the `bioinformer` program `Evolution` → `Phylogeny` to visualize the process of bootstrapping a phylogeny.

9

Sequence Variation and Molecular Evolution

> *Sequence variation is the currency of genetics; the central aim of all genetics is to correlate specific molecular variation with phenotypic changes.*
>
> *Aravinda Chakravarti [36]*

Sequence variation is detected through the comparison of DNA or protein sequences. In most cases, biologists compare sequences which are *related* by common ancestry, for instance those shown in Figure 9.1. Such sequences are called *homologous*.

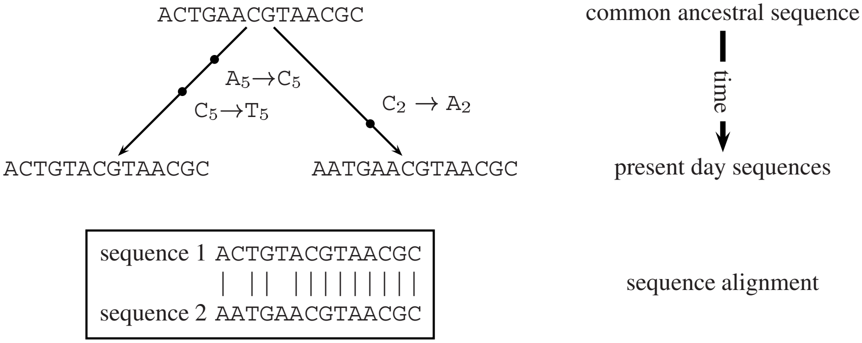

ACTGAACGTAACGC
common ancestral sequence
A5→C5
C5→T5
C2 → A2
time
ACTGTACGTAACGC
AATGAACGTAACGC
present day sequences
sequence 1 ACTGTACGTAACGC
sequence 2 AATGAACGTAACGC
sequence alignment

Fig. 9.1. Diverging sequences accumulate substitutions, here represented as filled circles along the branches of a genealogical tree. Substitutions lead to mismatches in the alignment (at positions 2 and 5). Note that there may be more substitutions than mismatches.

Note that this definition does not imply that homologous sequences are necessarily similar. Members of a set of homologous sequences may vary strongly in their similarity to each other, as was already shown in the alignment of seven globin sequences (Fig. 5.1). Homologous sequences are said to be *orthologous* if they are derived from a speciation event, such as the β-globin sequences from human and horse. Homologous sequences are *paralogous* if they are derived from a duplication event such as the α- and β-globins in vertebrates. While we may not be able to infer sequence similarity from the mere fact of homology, recently diverged sequences can safely be assumed to be similar. The converse, that similar sequences are homologous and hence have similar functions, is usually also true. There are exceptions to this rule, however. For example, later on in this chapter we are going to describe a gene thought to be associated with human language, *FOXP2*. Surprisingly, the highly conserved protein sequences encoded by this gene are more similar between chimpanzee and mouse than between chimpanzee and human.

It is one of the fundamental aims of molecular evolutionary biology to infer the type and time of past evolutionary events from the presently observable variability of molecular sequences. Such events cover a time scale which ranges from days, as in the evolution of strains of *E. coli*, to hundreds of millions of years, as in the evolution of the phylum of arthropods.

In this chapter we will introduce some basic terminology and expand on the evolutionary dimension of sequence comparison.

9.1 The Record of Past Events

When comparing homologous sequences, aligned positions may be occupied by identical characters (matches). From an evolutionary point of view and the assumption of parsimony, it is reasonable to conclude that matching characters are derived from the same character in a common ancestral sequence. On the other hand, substitutions of one nucleotide by another during evolution become manifest as mismatches in the alignment (Fig. 9.1). Insertions or deletions of nucleotides lead to alignment gaps.

At this point it is important to realize that the number of historical substitutions is greater or equal to the number of observable mismatches (Fig. 9.2). The reason

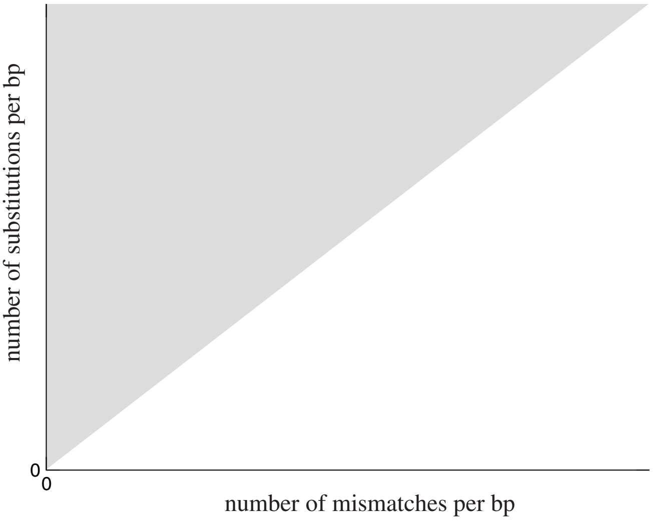


Fig. 9.2. The number of historical substitutions (gray shaded area) between two sequences is at least equal to, but may be greater than, the number of observable mismatches.

for this is that multiple substitutions at the same or at homologous sites cannot be directly observed from the present record (Fig. 9.1). This effect is more dramatic for distantly related sequences than for very closely related ones. Much effort in evolutionary modeling is spent on correcting for this effect, as discussed in the following sections.

9.2 Mutations and Substitutions

There are two fundamentally different ways in which one talks about DNA or protein *sequences*. First, one may refer to a particular sequence of a particular individual. For instance, a defense lawyer may state that "the DNA sequence of suspect X did not match the probe". Second, one may refer to a representative of a large class of equivalent sequences, as in the phrase "the sequence of the human genome was published in the year 2001". The sequence deposited in, say, the NCBI genome database GenBank *represents* a class of several billion human sequences, many of which do not even exist yet at the current time. Neither is it implied that the human genome in the database was derived from one particular existing individual.

A mutation is a heritable change in the nucleotide sequence of a chromosome. It takes place in a specific individual and will subsequently either vanish from or spread throughout the population. If it spreads throughout the population, the result is called a *substitution*. For instance, proline at position 4 in human α-globin has been substituted by alanine in horse at some time in the past (Fig. 5.1).

The number of mutations per nucleotide per individual is called the mutation rate and abbreviated μ. The substitution rate, abbreviated k, measures the fraction of those mutations, per unit time, which have spread throughout the entire population being studied. The substitution rate is also called the *rate of molecular evolution*. The two terms mutation and substitution are sometimes applied synonymously. This confusion is in part due to the following fundamental property of the two processes.

Consider a mutation that originates in a particular DNA sequence with mutation rate μ per generation. In a finite population (or species) of N diploid individuals, $2N\mu$ new mutations arise on average per generation. However, only a fraction of these new mutants survives and is passed on to descendant individuals in subsequent generations. Substitutions are exactly those mutations which will eventually be present in all individuals of a given population (or species). As explained in Chapter 10, under a neutral model of evolution, the probability for a newly arising mutation to be propagated to the entire population, i.e. to become a substitution, is $1/(2N)$. Therefore, one has the following fundamental relationship between the rate of substitution, k, and μ:

$$k = 2N\mu\frac{1}{2N} = \mu\,. \tag{9.1}$$

9.3 The Molecular Clock

Do substitutions occur at a clock-like pace, in other words, at a constant rate in time? If so, then such a *molecular clock* could be used to date evolutionary events, for instance the point in time at which two species diverged. The divergence time of two sequences should be proportional to the number of mismatches in a pairwise alignment, if the following three conditions hold: (i) mutations occur at a constant and known rate, (ii) the fraction of mutations which eventually become substitutions remains constant over time, and (iii) a monotonic relationship exists between the number of substitutions and the number of observable alignment mismatches.

Zuckerkandl and Pauling [265, 266] were the first to investigate in detail the relationship between the number of substitutions and divergence time. They compared globins and other protein sequences from various mammalian species and noted that the number of substitutions per unit time between two homologous sequences was roughly constant. Based on these results, Zuckerkandl and Pauling suggested that all proteins evolve at an essentially constant rate, known as the *molecular clock hypothesis*. The existence of a molecular clock on the level of DNA became one of the central concepts of the neutral theory of evolution, developed by Kimura and Ohta in the 1970s and 1980s [144, 140, 141]. The question of how general is the molecular clock for the evolution of DNA and proteins is still intensely debated today [28].

The seven globin sequences shown in Figure 5.1 are all derived from a "proto"-globin which existed some 1.5 billion years ago. Table 9.1 shows the number of mismatches in pairwise alignments of human α_1-globin with five other globins and their divergence times. Note that the percent mismatches at the amino acid level may be equal to, larger, or smaller than at the DNA level. Figure 9.3A shows a strikingly linear relationship between the mismatch percentage and the divergence time. For example, when comparing human α_1- and horse α-globins, one finds that 124 out of 141 amino acids, i.e. 87.9%, are identical and 12.1% of the aligned positions are

Table 9.1. Mismatch percentages between the coding sequence of human α_1-globin and five homologous globins[1].

	DNA[2]	AA[3]	time[4]
(1) human α_2	0.00	0.00	20
(2) chimp α_1	0.07	0.00	5
(3) horse α	13.3	12.1	80
(4) carp α	40.9	52.4	400
(5) human β	41.2	57.4	450

[1] using the program `align` [187] with substitution matrix BLOSUM50

[2] pairwise alignment of the cDNA sequences

[3] pairwise alignment of amino acid sequences

[4] approximate divergence time in million years

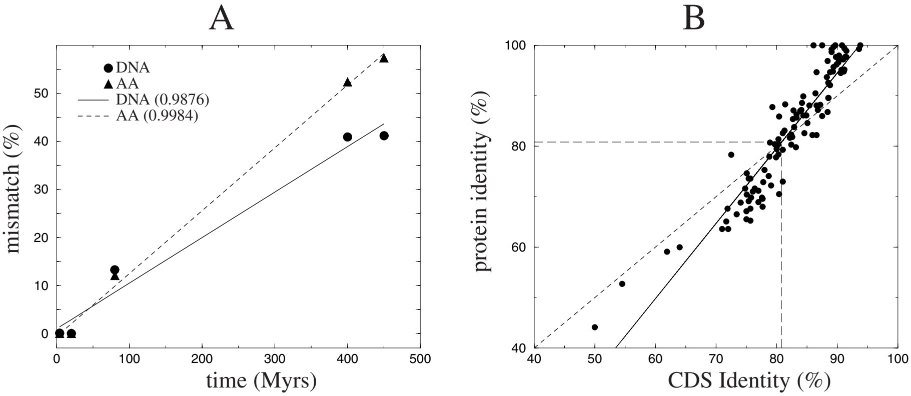


Fig. 9.3. A: Comparison of divergence time and percent mismatches from the globin example (Table 9.1) of DNA (circle) and the corresponding proteins (triangles). Only the coding part of the DNA is considered. Regression lines are plotted solid (DNA) or dashed (amino acids). Numbers in parentheses refer to the correlation coefficients. **B**: Comparison of percent identity in DNA (coding sequences, CDS) and protein alignments in a set of 110 orthologous genes from humans and rodents. The dashed line represents $x = y$. Its intersection with the regression line indicates an over-representation of synonymous differences in highly similar genes (identity $> 80\%$) and of non-synonymous differences in less similar genes (identity $< 80\%$).

mismatches. Human α_1- and β-globins align identically only in 63 out of 148 amino acids, i.e. there are 57.4% mismatches. This difference is best explained by the fact that α- and β-globins were derived from an ancient gene duplication which long predates (about 450 million years (Myrs) ago) the emergence of the mammalian lineages about 80 Myrs ago. On the other hand, α_1-globins and α_2-globins are present in human and apes, but not in horses. They are derived from a duplication which occurred about 20 Myrs ago. Despite their age, the coding sequences of these two genes are completely identical. In contrast, the relatively short divergence time of 5 Myrs between the orthologous human and chimpanzee α_1-globins was sufficient to accumulate some sequence divergence at the DNA level. This observation contradicts the molecular clock hypothesis. Is it therefore disproven altogether? To answer this question, we have to look in more detail at the ingredients of evolutionary models.

9.4 Explicit Models of Molecular Evolution

In the preceding section, only the percent of mismatches in pairwise alignments was compared to the divergence time. The molecular clock hypothesis, however, states a linear relationship between the rate of substitution and divergence time. To investigate this issue further, consider a particular position in a DNA sequence. It may be occupied by any of the four nucleotides, adenine (`A`), cytosine (`C`), guanine (`G`), or thymine (`T`). At the time of replication, a given nucleotide x is replaced by nucleotide y with probability p_{xy}. The replacement probabilities can be summarized in a state transition matrix of size 4 · 4. Remember that an empirical state transition matrix for amino acids was already shown in the context of the derivation of the amino acid substitution matrices (Table 2.2). The simplest model of DNA sequence evolution is that all possible state transitions are equally likely. The resulting matrix is shown in Equation (9.2). Only one parameter (μ) is needed in this model and therefore it is called the 1-parameter model, or, after its inventors, the Jukes-Cantor model [130].

$$M_1 = \begin{array}{c|cccc} & \mathtt{A} & \mathtt{C} & \mathtt{G} & \mathtt{T} \\ \hline \mathtt{A} & 1-3\mu & \mu & \mu & \mu \\ \mathtt{C} & \mu & 1-3\mu & \mu & \mu \\ \mathtt{G} & \mu & \mu & 1-3\mu & \mu \\ \mathtt{T} & \mu & \mu & \mu & 1-3\mu \end{array} \tag{9.2}$$

Repeating the process of replication with mutation infinitely many times, eventually any nucleotide occurs with equal probability at any given site. The state transition matrix has the following limiting property

$$\lim_{t\to\infty}(M_1)^t = \lim_{t\to\infty}\begin{pmatrix} z^+(t) & z^-(t) & z^-(t) & z^-(t) \\ z^-(t) & z^+(t) & z^-(t) & z^-(t) \\ z^-(t) & z^-(t) & z^+(t) & z^-(t) \\ z^-(t) & z^-(t) & z^-(t) & z^+(t) \end{pmatrix} = \begin{pmatrix} 1/4 & 1/4 & 1/4 & 1/4 \\ 1/4 & 1/4 & 1/4 & 1/4 \\ 1/4 & 1/4 & 1/4 & 1/4 \\ 1/4 & 1/4 & 1/4 & 1/4 \end{pmatrix} \tag{9.3}$$

where

$$z^+(t) = \frac{1}{4} + \frac{3}{4}(1-4\mu)^t$$

and

$$z^-(t) = \frac{1}{4} - \frac{1}{4}(1-4\mu)^t.$$

A slightly more involved, but also more realistic way of modeling the mutation process is captured in the so-called two-parameter or Kimura model. It distinguishes between nucleotide-transitions and nucleotide-transversions. A transversion is the replacement of a purine (`A` or `G`) by a pyrimidine (`C` or `T`). A transition is the replacement of a purine by a purine or of a pyrimidine by a pyrimidine. This distinction reflects biochemical differences between these two classes of nucleotides and the fact that transitions occur at a different rate than transversions. Two parameters are then needed to define the state transition matrix.

$$M_2 = \begin{array}{c|cccc} & \mathtt{A} & \mathtt{C} & \mathtt{G} & \mathtt{T} \\ \hline \mathtt{A} & 1-\mu_1-2\mu_2 & \mu_2 & \mu_1 & \mu_2 \\ \mathtt{C} & \mu_2 & 1-\mu_1-2\mu_2 & \mu_2 & \mu_1 \\ \mathtt{G} & \mu_1 & \mu_2 & 1-\mu_1-2\mu_2 & \mu_2 \\ \mathtt{T} & \mu_2 & \mu_1 & \mu_2 & 1-\mu_1-2\mu_2 \end{array} \tag{9.4}$$

It would be straightforward to set up a model with the maximal number of 12 parameters. However, it would be difficult to determine numerical estimates for all these parameters from experimental data and to work with such a model in practice.

9.5 Estimating Evolutionary Rates

With the help of an explicit model of sequence evolution, an estimate of the substitution rate, k, can be obtained. Consider two sequences as depicted in Figure 9.1. Looking at a particular position and starting from the ancestral sequence (time $t = 0$), we determine the probability $P_{\text{id}}(t)$ that in the descendant sequences this position will be occupied by identical nucleotides at time t. We calculate this probability recursively and consider the possible changes when going from one generation ($t - 1$) to the next (t). Two cases have to be distinguished. In the first case, both nucleotides were identical in generation $t - 1$. The probability for this is $P_{\text{id}}(t - 1)$. In addition, no mutation occurred on any of the two branches between generations $t - 1$ and t. Under the Jukes-Cantor model, the probability for the latter event is $(1 - 3\mu)^2$. In the second case, the site was occupied by two different nucleotides in generation $t - 1$ and a mutation occurred, which made them identical in generation t. The probability for this combined event under the Jukes-Cantor model is $2\mu\ (1 - P_{\text{id}}(t - 1))$, because there are two ways in which two distinct nucleotides can become identical. Taken together, the desired probability $P_{\text{id}}(t)$ can be expressed in terms of $P_{\text{id}}(t-1)$ as

$$P_{\text{id}}(t) = (1 - 3\mu)^2\ P_{\text{id}}(t - 1) + 2\ \mu\ (1 - P_{\text{id}}(t - 1)).$$

For small mutation probabilities ($0 < \mu << 1$), one has $\mu^2 << \mu$. Therefore, terms involving the squared mutation probability may be neglected to obtain the approximation

$$P_{\text{id}}(t) \approx (1 - 8\mu)P_{\text{id}}(t - 1) + 2\mu\,. \tag{9.5}$$

Abbreviating the difference $P_{\text{id}}(t) - P_{\text{id}}(t-1)$ by $\Delta P_{\text{id}}(t)$, one derives the following difference equation

$$\frac{\Delta P_{\text{id}}(t)}{\Delta t} = \frac{P_{\text{id}}(t) - P_{\text{id}}(t - 1)}{t - (t - 1)} = 2\mu(1 - 4P_{\text{id}}(t - 1))\,. \tag{9.6}$$

When treating time as a continuous variable and re-interpreting μ as a mutation *rate* per unit time, the corresponding differential equation reads

$$\frac{dP_{\text{id}}(t)}{dt} = 2\mu(1 - 4P_{\text{id}}(t))\,. \tag{9.7}$$

This is an ordinary linear differential equation. The initial condition $P_{\text{id}}(0) = 1$ derives from the fact that nucleotides are identical at time $t = 0$. The solution therefore is

$$P_{\text{id}}(t) = \frac{1}{4}(1 + 3e^{-8\mu t})\,. \tag{9.8}$$

The complementary probability of observing two different nucleotides after divergence for t generations is

$$P_{\text{diff}}(t) = 1 - P_{\text{id}}(t) = \frac{3}{4}(1 - e^{-8\mu t})\,. \tag{9.9}$$

Equation (9.9) represents the probability of a mismatch at one particular site in a (gap-free) alignment. Assuming that sites evolve independently, this probability is identical to the fraction of mismatches in the alignment. Let this fraction be denoted by D/L, where D is the absolute count of mismatches and L is the number of aligned positions. By inserting D/L on the lefthand side of Equation (9.9), solving for $8\mu t$, and multiplying by $3/4$, we obtain an estimate of the compound variable

$$6\mu t = -\frac{3}{4}\log(1 - \frac{4}{3}(D/L)) \,. \tag{9.10}$$

We now switch our point of view and re-interpret the above two-sequence genealogy as a two-species genealogy. Because of the equality of mutation and substitution rates under neutrality, stated in Equation (9.1), we only have to replace the mutation rate by the substitution rate. In the Jukes-Cantor model, the substitution rate per unit time is $k = 3\mu$. Therefore, we obtain

$$2\,k\,t = -\frac{3}{4}\log(1 - \frac{4}{3}(D/L)) \,. \tag{9.11}$$

There are two points to note about Equation (9.11). First, it relates the observable quantity D/L to the unobservable quantities k and t. Second, the product $2kt$ is the expected number of substitutions per site on a two-branch genealogy with divergence time t. This can be seen as follows. Assuming a molecular clock, substitutions accumulate independently and with a constant rate. Since substitutions are rare events they are well described by a Poisson-distributed random variable K with parameter k. Given a divergence time t, the total branch length in the genealogy is $2t$ and the expected number of substitutions per site is given by

$$E(K(t)) = 2\,k\,t \,. \tag{9.12}$$

It is now easy to combine Equations (9.11) and (9.12) to obtain

$$E(K(t)) = -\frac{3}{4}\log(1 - \frac{4}{3}(D/L)) \,. \tag{9.13}$$

Figure 9.4 shows a plot of $E(K(t))$ as a function of alignment mismatches, D/L. Note, however, that the actual value of $K(t)$ may be different for different sites, and that the righthand side of Equation (9.13) is an estimator for the mean of the random variable $K(t)$. This estimate will be closer to its true value if many sites are considered. One can derive an expression for the variance, V, of this estimate. It depends on the number of sites compared, i.e. the sequence length L:

$$V(E(K(t))) = \frac{(D/L)(1 - (D/L))}{L(1 - \frac{4}{3}(D/L))^2} \,. \tag{9.14}$$

If L increases the variance tends to zero. The variance has a singularity for $(D/L) = 0.75$, i.e. the variance of the estimate of $E(K)$ is infinitely large when two random sequences are compared (Fig. 9.4).

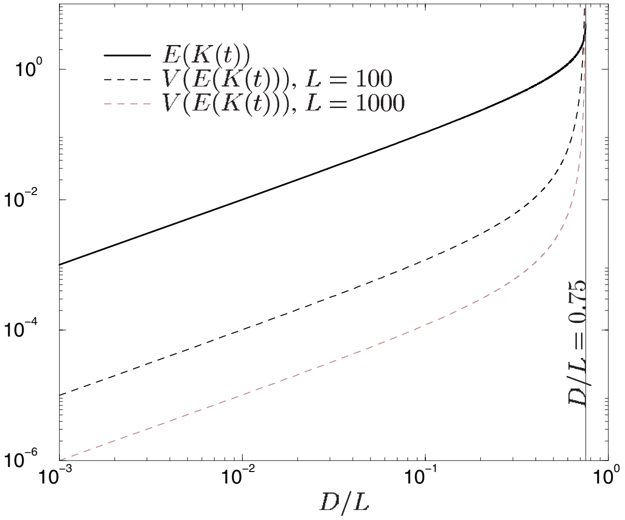


Fig. 9.4. Logarithmic plot of the expected number of substitutions, $E(K(t))$, *vs.* the number of mismatches per nucleotide, D/L (solid black). In random sequences, $D/L = 0.75$. Dashed lines indicate the variance of the estimate of $E(K(t))$.

Equation (9.11) contains neither t nor k separately. In particular, it is not possible to derive separate estimates for t or k without further information. In order to calibrate the molecular clock, the relationship between sequence divergence and actual divergence time has to be known at least for one sequence pair from an independent source. Paleontological data is often used for this purpose. Given such a calibration, the divergence times for arbitrary sequence pairs may be inferred. However, in practice there are important caveats to be considered. For instance, the rate of evolution may not be constant over time or sequences may not evolve according to a neutral model. One way to deal with the latter is to take into account possible selective constraints in the process of evolution and to distinguish between synonymous and non-synonymous substitution rates when comparing protein coding sequences.

9.6 Coding Sequences: Synonymous and Non-Synonymous Substitutions

In protein-coding DNA, substitutions are grouped into *synonymous* substitutions, which leave the encoded amino acid unchanged, and *non-synonymous* substitutions, which also change the encoded amino-acid. Consider an alignment of two protein-coding sequences (Fig. 9.5). In order to compute the implied substitution rate, we first determine the total numbers of **s**ynonymous (L_s) and of non-synonymous (L_a, **a**mino-acid-changing) sites. Next, we count the synonymous (D_s) and non-synonymous (D_a) mismatches. In analogy to Equation (9.13), one can then derive

K_s and K_a, the number of synonymous and non-synonymous substitutions per site, respectively. Under the Jukes-Cantor model, these formulas are

$$E(K_s) = -\frac{3}{4}\log(1 - \frac{4}{3}(D_s/L_s)) \text{ and} \tag{9.15}$$

$$E(K_a) = -\frac{3}{4}\log(1 - \frac{4}{3}(D_a/L_a))\,. \tag{9.16}$$

In order to apply these equations, one first needs to obtain L_s and L_a. Consider sequence 1 in Figure 9.5 and assume that the grouping of nucleotide triplets reflects the reading frame (i.e. frame 0 in this case). For any site j let $I(j)$ denote the possible number of synonymous changes at this site. $I(j)$ can take values 0, 1, 2, or 3 and can be read from the genetic code table (Table C.4). In sequence 1, we start with the first position of the `Phe` codon, which, like most first and second codon positions, allows no synonymous changes, i.e. $I(1) = 0$. At the second codon position we get the same result ($I(2) = 0$), but at the third position $I(3) = 1$. Switching to the next triplet encoding `Thr`, we find at its first position $I(4) = 0$ and so on. Any site j is counted as $I(j)/3$-synonymous and as $(3 - I(j))/3$-non-synonymous. In the example, the first site of sequence 1 is therefore 1-non-synonymous, the second site also, and the third site is $1/3$-synonymous and $2/3$-non-synonymous. Generally, the total number of synonymous sites in sequence 1, $L_s^{(1)}$, in a (gap-free) alignment of length L is

```
           Phe Thr Val Val Asn
sequence 1 TTT ACT GTC GTC AAT
           ::: ::: ::  : : :
sequence 2 TTT ACT GTT GCC ACG
           Phe Thr Val Ala Thr
```

Fig. 9.5. Alignment of two coding sequences.

$$L_s^{(1)} = \sum_{j=1}^{L} I(j)/3$$

and the total number of non-synonymous sites is

$$L_a^{(1)} = \sum_{j=1}^{L} (3 - I(j))/3\,.$$

The total number of synonymous (L_s) and non-synonymous (L_a) sites in the complete alignment is the average of the numbers from sequence 1 and sequence 2

$$L_s = \frac{L_s^{(1)} + L_s^{(2)}}{2}$$

and

$$L_a = \frac{L_a^{(1)} + L_a^{(2)}}{2},$$

respectively. In the above example, $L_s^{(1)} = 11/3$, $L_s^{(2)} = 13/3$, and $L_s = 4$. For the non-synonymous sites one has $L_a^{(1)} = 34/3$, $L_a^{(2)} = 32/3$, and $L_a = 11$.

Next, the number of synonymous and non-synonymous mismatches between sequence 1 and sequence 2, D_s and D_a, need to be counted. There may be one, two, or three differences in a pair of codons. If there is only one mismatch, it is counted as one synonymous or one non-synonymous difference depending on whether the amino acid is conserved between the sequences or not. For example, the third position in codon 3 in Figure 9.5 carries one synonymous difference, the second position in codon 4 carries one non-synonymous difference. If there are two or three differences in a codon pair, such as in the last codon of the example, one needs to consider the possible pathways which can produce the observed constellation. Depending on the pathway chosen, different numbers of synonymous and non-synonymous changes may be counted. For example, converting codon `AAT` to `ACG` requires at least two changes, one from `A` to `C` and one from `T` to `G`. The two possible pathways are:

$$\begin{aligned}
&\text{pathway 1: } \texttt{AAT}(\text{Asn}) \xrightarrow{\text{1 non-syn.}} \texttt{ACT}(\text{Thr}) \xrightarrow{\text{1 syn.}} \texttt{ACG}(\text{Thr}),\\
&\text{pathway 2: } \texttt{AAT}(\text{Asn}) \xrightarrow{\text{1 non-syn.}} \texttt{AAG}(\text{Lys}) \xrightarrow{\text{1 non-syn.}} \texttt{ACG}(\text{Thr}).
\end{aligned}$$

Evidently, the number of non-synonymous and synonymous changes depends on which codon position is changed first. This leaves the problem of choosing between the different pathways. One possibility is to consider all of them, but to apply a certain weighting scheme. The simplest weighting scheme assigns uniform weights. Accordingly, D_s and D_a are the arithmetic average of synonymous and non-synonymous mismatches over all possible pathways. This strategy was suggested by Pamilo and Bianchi [195] and is implemented for example in the program `DIVERGE`, which is part of the Wisconsin software package [86]. According to the Pamilo and Bianchi algorithm, one gets for the above example $D_a = 1/2 \cdot 1 + 1/2 \cdot 2 = 1.5$ and $D_s = 1/2 \cdot 1 + 1/2 \cdot 0 = 0.5$. Thus, the numbers of substitutions per site are $K_a = 0.1505$ and $K_s = 0.1367$. The ratio is $K_a/K_s = 1.101$. This ratio can be used as a test statistic to detect deviations from a neutral model of evolution. In the absence of selection the ratio should be close to one. Values smaller than one are indicative of purifying selection (the turn-over of amino acids is slower than expected under neutrality) while values larger than one are indicative of strong positive or diversifying selection (turn-over of amino acids is faster than expected). Such tests are implemented, for instance, in the software PAML (`abacus.gene.ucl.ac.uk/software/paml.html`).

9.7 Substitutions in Globin Sequences

Table 9.2 shows the values of K_a and K_s obtained with the program `dists1` [127] for the globin coding sequences. Linear regression yields correlation coefficients of 0.883 and 0.997 for the K_s and K_a data, respectively (Fig. 9.6A). Compared to the correlation coefficient of 0.988 for the DNA data shown in Figure 9.3A, it appears that the synonymous substitutions have a less clock-like behavior. In particular, the K_s-value for human β-globin is puzzling. On the other hand, the K_a-values cluster very tightly around the regression line and produce a similar correlation coefficient as the amino acid data (0.998) in Figure 9.3A.

Table 9.2. Number of substitutions per bp between human α_1-globin and five other globins[1].

	K_s (JC)[2]	K_s (K)	K_a (JC)	K_a (K)	K_a/K_s (JC)	K_a/K_s (K)
human α_2	0.0000	0.0000	0.0000	0.0000	–	–
chimp α_1	0.0256	0.0256	0.0000	0.0000	0.0	0.0
horse α	0.2840	0.2920	0.0987	0.0987	0.3475	0.3380
carp α	1.4700	1.6100	0.4990	0.4990	0.3395	0.3099
human β	0.7840	0.8600	0.5110	0.5120	0.6518	0.5953

[1] only the coding sequences (CDS) of the genes are considered

[2] calculations are performed with the program `dists1` [127]; JC refers to the Jukes-Cantor model, K to the Kimura model

The generally observed low ratio of K_a to K_s in the comparison of human α_1-globin with its orthologues in chimpanzee, horse, and carp suggest that their evolution is subject to purifying selection (Fig. 9.7A). In the comparison of human α_1-globin with its paralogue β-globin one finds less evidence for purifying selection. A reasonable biological explanation for this is that duplicated genes may be less subject to functional constraints and may therefore accumulate non-synonymous changes more rapidly. On the other hand, the coding sequences of the recently duplicated human paralogues α_1- and α_2-globin are completely identical, leading to $K_a = K_s = 0$. Differences in the two genes are limited to the non-coding part. For a more rigorous analysis, it is advisable not to lump paralogues and orthologues together. When the data point for the human β-globin gene is removed from the regression analysis, the correlation coefficient for the K_s data increases to the value of 0.9987 (gray line in Figure 9.6A). Instead of plotting K_a/K_s *vs.* divergence time, one may be interested in identifying regions under selection within a gene. Figure 9.7B shows the result of a sliding window analysis of the ratio K_a/K_s for the human/horse comparison. The window size in this example was set to 120 bp and the step size to 30 bp. The resulting graph oscillates around its mean value of $K_a/K_s = 0.338$ (Table 9.2). There is a minimum of K_a/K_s around position 300, indicating high conservation of the encoded amino acids in this region. This part of the protein corresponds to the heme-pocket. However, the oscillations are in part also due to random fluctuations around the mean value as described by the variance of the

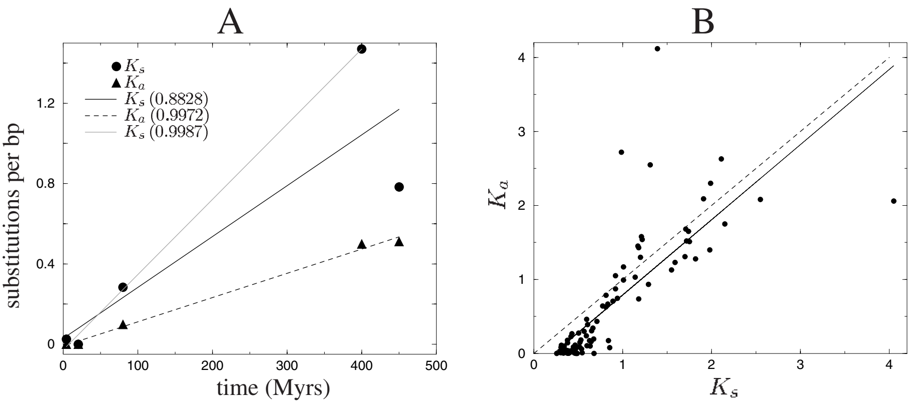


Fig. 9.6. **A**: Number of substitutions per bp for the globin data (Table 9.1) inferred from the alignments and using the Jukes-Cantor model as implemented in the program `dists1` [127]. Regression lines for the full data set (five comparisons) are in black. The regression line in gray is obtained when the comparison of human α_1-globin and β-globin is excluded from the analysis. The correlation coefficients are quoted in parentheses. **B**: Plot of K_a *vs.* K_s for a set of 100 rodent-human orthologous genes. The solid line represents the linear regression line, the dashed line the diagonal $x = y$.

substitution rate in Equation (9.14). To distinguish random fluctuations of the ratio K_a/K_s from deviations caused by selection, a rigorous statistical test is needed.

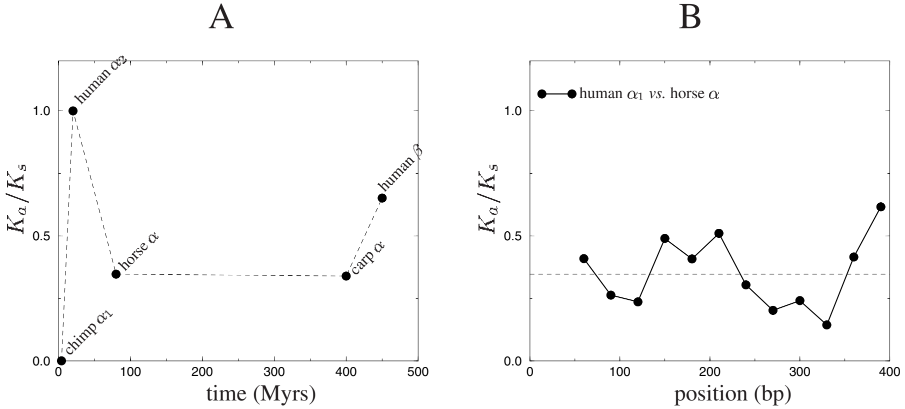


Fig. 9.7. **A**: Plot of the ratio of non-synonymous to synonymous substitutions (K_a/K_s) for the five globin comparisons (Table 9.1). Horse and carp α-globin, which are orthologous to human α_1-globin, have almost identical values. The paralogues human α_2- and β-globin differ strongly. **B**: Sliding window analysis of the human α_1-globin *vs.* horse α-globin comparison. X-axis: position along the coding sequence. Data points were taken in a window of 120 bp and step size 30 bp. K_a and K_s values were calculated with the program `dists1` [127].

9.8 Applications of K_a/K_s

Computations of relative synonymous and non-synonymous substitution rates are usually carried out in order to detect selection on protein-coding genes. Such investigations may either concentrate on a single gene of interest, or—given the completion of suitable genome sequencing projects—survey entire proteomes.

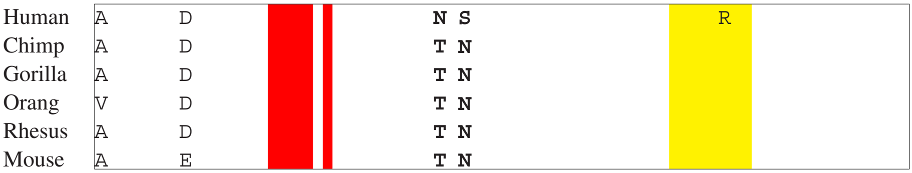
9.8.1 A Language Gene?

Language is an exclusively human trait. This assertion may surprise some readers, as reports of the allegedly fabulous linguistic abilities of intelligent animals such as chimpanzees and dolphins periodically capture the public imagination. However, we can safely state that no animal is capable of the kind of rule-based, open-ended sentence production that three-year old children have mastered to a large extent. In contrast, no human population without language has ever been discovered. In the same highly specialized way in which some animals can fly and others can swim, we can talk [203].

There is both a "nature" as well as a "nurture" aspect to this ability. Any newborn infant discovers the grammatical rules of whichever language he finds himself surrounded by. However, a childhood spent without verbal interaction leads to a lifetime of linguistic disability [203]. The "nature" part of language acquisition must be grounded in genetics, but until the early 1990s there was no suitable model to study this in. This changed with the publication in 1990 of a report on a family with a dominantly inherited severe speech and language disorder [138]. Affected members of this family, known as KE, are deficient in the coordinated movements required for speech; for example, they have difficulty pronouncing somewhat complicated words such as "rhinoceros". However, there was immediate debate about the precise nature of the disorder. One opinion was that it consisted of a blindness to grammatical features such as case markers, etc. Others held that the disorder was due to problems in language production due to impaired control of facial movements. Finally, it was argued that the disorder was not attributable to a single cause. This uncertainty about the interpretation of the disorder's phenotype has persisted to the present day [138].

In contrast, the gene underlying the disorder was discovered in 2001. Forkhead box protein 2, or *FOXP2*, belongs to a large class of transcription factors known as winged-helix or forkhead proteins. It is located on human chromosome 7 (7q31) and its dominant splice form consists of 715 amino acids. Members of the KE family are characterized by the single amino acid exchange $\texttt{Arg}_{553} \rightarrow \texttt{His}_{553}$. Disruption of *FOXP2* by a translocational breakpoint in an individual unrelated to KE results in the same speech disorder observed in the affected KE family members [138].

There is a great difference between the disruption of a function and its construction. Clearly, changes in *FOXP2* can lead to an impairment of linguistic ability, but so can many other mutations. The question is, how central is *FOXP2* for the construction of our talkative phenotype? Presumably language has selective value and this premise led to a study of the evolution of *FOXP2* in the wake of its identification [63]. Figure 9.8 displays an alignment of FOXP2 proteins from human, chimpanzee, gorilla, orangutan, rhesus, and mouse. The proteins are characterized by two polyglutamine stretches that are variable. Since this variability does not co-segregate with any known phenotype, it is disregarded in the analysis. However, outside these two regions, only four amino acid replacements have occurred since the divergence of mice and monkeys 70 million years ago. Humans differ from the mouse sequence at three positions, and a fourth substitution differentiates the orangutan. This places FOXP2 among the 5% most highly conserved mammalian proteins [63].



Human	A	D	**N**	**S**	R
Chimp	A	D	**T**	**N**	
Gorilla	A	D	**T**	**N**	
Orang	V	D	**T**	**N**	
Rhesus	A	D	**T**	**N**	
Mouse	A	E	**T**	**N**	

Fig. 9.8. Alignment of FOXP2 sequences. The four amino acid substitutions outside of the poly-glutamine regions are marked by columns of residues. The two human-specific substitutions are shown in bold. The forkhead domain contains the arginine (R553) that is mutated to a histidine in affected members of the KE family. Adapted from [63].

If both the synonymous as well as the non-synonymous polymorphisms in *FOXP2* are plotted on the phylogeny of the respective animals, a striking pattern emerges (Fig. 9.9): two of the three amino acid substitutions that separate mice and humans are located on the human lineage. The human population is fixed for these two amino acid replacements. Moreover, along the human lineage only non-synonymous substitutions have taken place. The null hypothesis that there is no human-specific K_a/K_s ratio could be rejected with high significance ($P < 10^{-3}$). In other words, positive selection might have driven the fixation of the human *FOXP2* variant [63]. Notice that it was the combination of polymorphism analysis and phylogeny reconstruction that lead to this conclusion. Calculation of substitution rates for *FOXP2* irrespective of phylogeny would have simply led to the detection of very low K_a/K_s ratios.

9.8.2 Selection in the Human Genome

Estimating synonymous and non-synonymous substitution rates is also a key method in comparative genome analysis. In a search for positively selected genes in the genomes of humans and chimpanzees, 13,731 annotated genes from humans were compared to their chimpanzee orthologues [190]. Based on their K_a/K_s ratios, the genes with the strongest signal of positive selection included proteins involved in sensory perception and immune defense (Table 9.3). Perhaps surprisingly, the list of genes under strong positive selection also included genes involved in tumor suppression and apoptosis. Some of the genes for spermatogenesis also displayed strong positive selection. Given the great interest in the molecular basis of the difference in cognitive abilities between humans and chimpanzees, it was perhaps disappointing

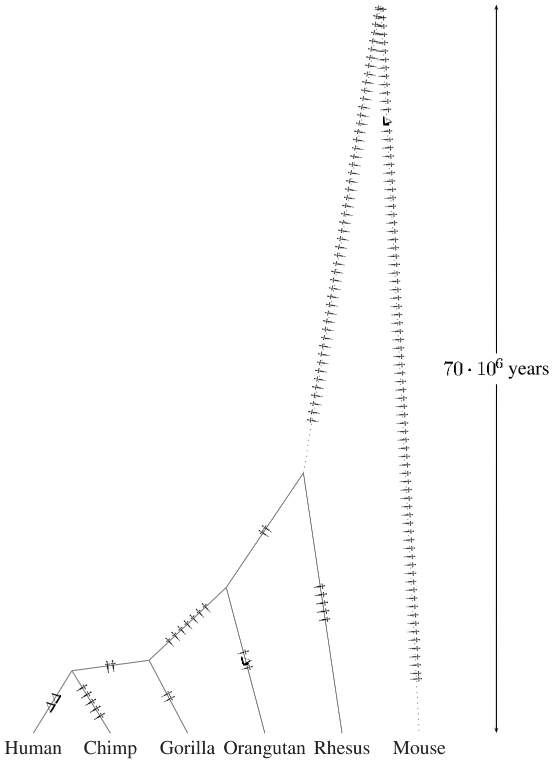


Fig. 9.9. Synonymous (†) and non-synonymous (Δ) substitutions in the *FOXP2* gene plotted on the phylogeny of six mammals. Note that the distribution of polymorphisms along the dotted branches connecting the most recent common ancestor of the clade to mouse and the most recent ancestor of monkeys is unknown in the absence of an outgroup. Adapted from [63].

to learn that genes with maximal expression in the brain showed little or no evidence for positive selection.

9.9 Summary

In this chapter we studied the relationship between sequence variation and the history of sequences. The concept of evolutionary relatedness of two sequences is connected to the idea of a common ancestral sequence and is visualized by a tree with two branches. The most widely used method to measure evolutionary relatedness is based on an alignment. Starting from this, the number of evolutionary events separating two sequences may be inferred. Here we concentrated on point mutations and showed how the number of substitutions can be computed. As far as the evolution of protein coding sequences is concerned, the number of substitutions must be disassembled

Table 9.3. The top 12 genes showing evidence for positive selection in a genome wide comparison of human and chimpanzee orthologous genes. D_a and D_s: non-synonymous and synonymous substitutions between humans and chimps, respectively; S_a and S_s: non-synonymous and synonymous segregating sites in humans, respectively; LR: likelihood ratio from the likelihood ratio test of $K_a/K_s = 1$ *vs.* $K_a/K_s > 1$ in the human-chimp alignment. Adapted from a comparative analysis of all genes in human and chimp by Nielsen and colleagues [190].

Gene Name	Function	Human-Chimp Divergence		Human Polymorphism		LR
		D_a	D_s	S_a	S_s	
PRM1	Substitutes for histones in sperm	9	0	0	2	10.1
CMRF35H	Leukocyte membrane antigen	13	0	0	0	9.3
DGAT2L1	Fatty acid synthesis (presumed)	10	1	2	0	6.6
FLJ46156	Unknown	10	1	4	3	6.4
USP26	Testis-specific expression	11	0	1	0	6.2
C15orf2	Testis-specific expression	18	2	12	4	6.1
ABHD1	Unknown	6	0	4	1	5.8
SCML1	Transcriptional repressor, embryonic development (hypot.)	15	1	0	0	5.7
OR2W1	Olfactory receptor	8	0	2	1	5.7
LOC389458	Unknown	8	0	1	0	5.5
APOBEC3F	Antiretroviral factor	11	0	2	1	5.5
MS4A12	Unknown	8	0	1	1	5.4

into two components—synonymous and non-synonymous changes. By considering the ratio of synonymous to non-synonymous substitutions, selection can be detected.

9.10 Further Reading

Wen-Hsiung Li has written a comprehensive textbook on molecular evolution [168].

9.11 Exercises

9.1. Why is the maximal number of free parameters in a DNA substitution model equal to twelve?

9.2. Verify the approximation in Equation (10.9).

9.3. What is the expected mismatch percentage, if two random DNA sequences are compared?

9.4. Verify the limiting property of matrix M_1 in Equation (9.3).

9.5. Derive $K(t)$ for the two-parameter model.

9.6. How many possible pathways exist if a pair of aligned codons differs at all three positions?

9.7. The Jukes-Cantor as well as the Kimura model assume that different sites in a DNA sequence evolve independently. What must be changed if this assumption is dropped?

10

Genes in Populations: Forward in Time

The very small range of selective intensity in which a factor may be regarded as effectively neutral suggests that such a condition must in general be extremely transient.
Ronald A. Fisher [76, p. 95]

The principal evolutionary mechanism in the origin of species must thus be an essentially nonadaptive one.
Sewall Wright [263, p. 364]

Genetic diversity is ubiquitous and some of the best-known examples in humans include the different sex chromosomes, the different blood groups, and the presence or absence of genetic diseases such as cystic fibrosis. In fact, except for monozygotic twins, all humans are genetically distinct. At the molecular level this observation corresponds to the knowledge that no two humans have the same genome sequence. Although there is evidence for abundant large scale genetic variation between humans [126], a major fraction of the hitherto studied human variation concerns *single nucleotide polymorphisms* (SNPs). When comparing two homologous sequences from humans, there is approximately one such SNP per kilobase [244, 11]. However, this number can vary widely between different genomic regions, between different populations and it can be very different in other species.

As Gillespie has put it, it is the "great obsession" of population geneticists to account for the causes and consequences of genetic diversity found in natural populations [87, p. 4]. A *population* is a reproductive community of organisms belonging to the same species. Figure 10.1 illustrates that a genealogical tree of organisms taken from the same species is generally much shallower than a species tree. As a consequence, the number of mismatches and indels found in intra-specific alignments is generally much smaller than in inter-species comparisons. This, in turn, has important implications for the way we interpret and model sequences and their polymorphisms.

10.1 Polymorphism and Genetic Diversity

In Chapter 9 we have stressed the fact that new mutations originate as a single copy. The chromosome carrying the lone new variant or *allele* can be passed on to multiple descendants in subsequent generations. In this way, the frequencies of the novel and of the previously existing allele, also called *wild-type* allele, may change. Depending on factors such as chance and reproductive success of their carriers, both alleles will

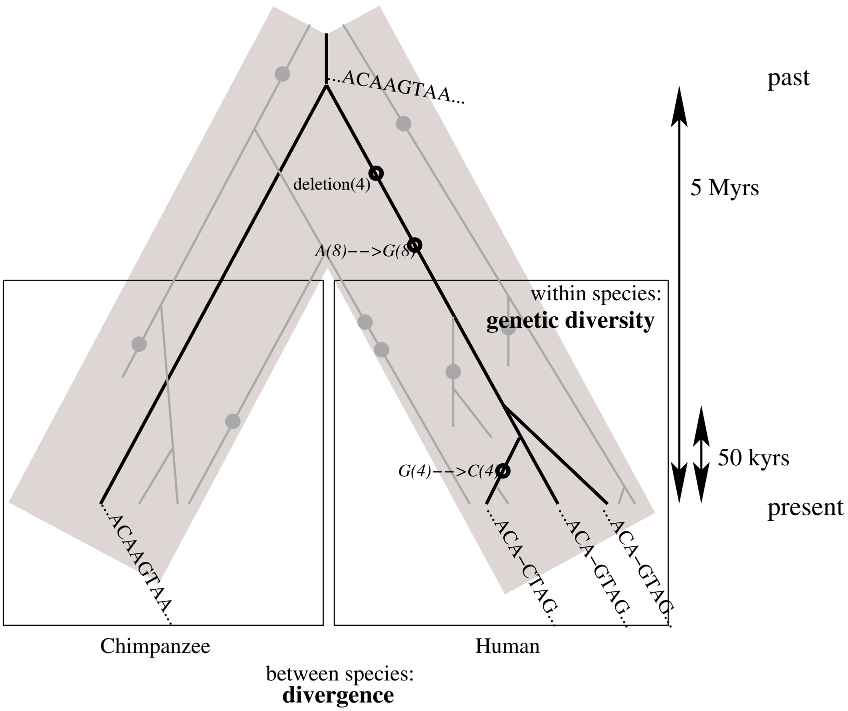

...ACAAGTAA...
deletion(4)
A(8)−−>G(8)
within species:
genetic diversity
G(4)−−>C(4)
past
5 Myrs
50 kyrs
present
...ACAAGTAA...
...ACA−CTAG...
...ACA−GTAG...
...ACA−GTAG...
Chimpanzee
Human
between species:
divergence

Fig. 10.1. A genealogical tree of DNA sequences. The black lines indicate the genealogical history of a sample of four homologous DNA sequences. While the most recent common ancestor of human and chimpanzee existed about 5 million years ago, the last common ancestor of the three human sequences belonged to an individual who lived approximately 50,000 years ago. The gray lines indicate lineages which are not present in the sample or which became extinct. The amount of variation is generally much smaller within species (genetic diversity) than between species (divergence).

be present for some time in the population. However, eventually only one allele will survive and there is some chance that the new allele will have *substituted* the previous wild-type allele (Fig. 10.2). The fluctuation of allele frequencies due to chance and random fixation of one or the other allele is called *random genetic drift*. The simultaneous presence in a population of two or more alleles at a defined position in the genome is called a *polymorphism*. A single nucleotide polymorphism, or *SNP*, refers to a polymorphism at a single nucleotide site in the genome. Its cause is a point-mutation, i.e. a single base exchange. The vast majority of SNPs are *bi-allelic*. An example is the $\mathtt{G}_4/\mathtt{C}_4$ polymorphism in the human lineage in Figure 10.1, with the two alleles `G` and `C` at position 4. Another type of frequently occurring polymorphism is due to replication slippage and becomes manifest as *length polymorphism*,

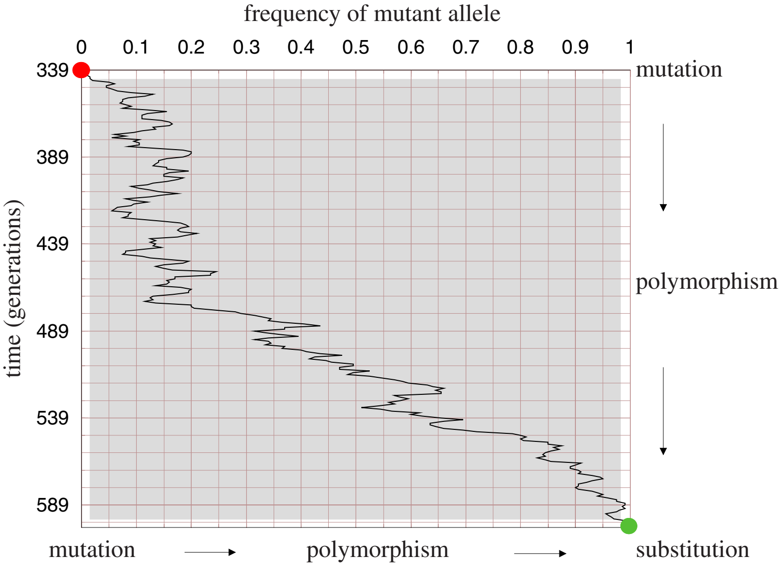


Fig. 10.2. Polymorphisms originate as mutations and may turn into substitutions due to random genetic drift. A computer simulation was performed under a two-allele neutral Wright-Fisher model with parameters $N = 100$ and $\mu = 0.001$. The trajectory shown and originating at generation $t = 339$ eventually reaches frequency $f = 1$. In practice, the term "polymorphism" is often used only for those cases in which the frequency of the minor allele is $f \geq 0.01$; this corresponds to the area shaded in gray.

in particular in tandem repetitive DNA. Stretches of tandemly repeated DNA are also called *micro-* or *minisatellites*, depending on the size of the repeat unit (Fig. 10.3A). Typical microsatellites have repeat units of length 2-3 bp, minisatellites of up to several 100 bp. Microsatellites usually are multi-allelic and constitute important genetic markers because of their high variability between organisms.

The average amount of polymorphisms measured within a population and across some genomic region is called *genetic diversity*. The first study of genetic diversity in *D. melanogaster* at the nucleotide sequence level was published by Kreitman in 1983 [152]. He sequenced a stretch of 2,721 bp at the *Adh* locus in a sample of 11 flies. Kreitman found 43 single nucleotide polymorphisms among the 2,721 sites. In addition, his data set contained six indels, yielding a total of 49 variable sites. Two out of the 11 sequences were identical, of the remaining nine each occurred only once in the sample (Table 10.1). Notice, however, that *Drosophila* strain Wa-F is distinguished from strain Af-F by a single nucleotide in the length of the insertion ∇_3 located in the 3' untranslated region. In our subsequent treatment of this data set we will ignore this polymorphism and say that it contains a total of nine distinct alleles.

We can also combine the analysis of genetic variation found within a population or species with that found between species. For example, we may compare two se-

Table 10.1. Polymorphism at the *Adh* locus of *Drosophila melanogaster*. The boxed strains are identical (ignoring the length polymorphism at insertion ∇_3). The *F* and *S* in the strain designations refer to the *fast* and *slow* allozyme variants of the *Adh* enzyme. ∇/Δ: insertion/deletions numbered from left to right; numbers in the insertion/deletion columns refer to length differences compared to the consensus sequence. The underlined polymorphism in exon 4 causes the Thr/Lys amino acid replacement that underlies the fast/slow allozyme polymorphism. Data taken from [152].

Strain	5' Flanking	Exon 1	Intron 1	Exon 2	Intron 2	Exon 3	Intron 3	Exon 4, translated	3' Untranslated	3' Flanking
	CCG		CAATATGGG∇_1C∇_2GC	T	AC	CCCC	GGAAT	CTCC$\underline{\text{A}}$CTAG	A∇_3C	AGC∇_4C∇_5TΔ_6
Wa-S	...		.....AT..	.	..	TT.A	CA.TA	AC.......	. . .	.... .Δ
Fl-1S	..C		....	.	..	TT.A	CA.TA	AC.......	. . .	.... .Δ
Af-S	...		....	.	..	....	.....	.........A	. . .	..T ∇ . 1 A .
Fr-S	...		....	.	GT	....	.....	.........A	. -1 .	TA. . . . . .
Fl-2S	...		AG...A.TC . . . .A	G	GT	....	.....	....	C 3 .	....
Ja-S	..C		....	G	..	....	.....	...T.T.CA	C 4 .	.... . T . . .
Fl-F	..C		....	G	..	....	.....	..GTCTCC.	C 4 .	....
Fr-F	TGC		AG...A.TC ∇ G ∇ ..	G	..	....	.....	..GTCTCC.	C 4 G	....
Wa-F	TGC		AG...A.TC ∇ G ∇ ..	G	..	....	.....	..GTCTCC.	C 4 G	....
Af-F	TGC		AG...A.TC ∇ G ∇ ..	G	..	....	.....	..GTCTCC.	C 5 G	....
Ja-F	TGC		AGGGGA... ∇ . . T.	G	..	..A.	..G..	..GTCTCC.	C 4 .	.... . . -1 . .
Polym. Sites	3	0	12	1	2	4	5	9	2	5
Nucleotides[1]	63	87	690	99	65	405	70	264	178	767
% polymorphic	4.7	0	1.7	1.0	3.1	1.0	7.1	3.5	1.1	0.6

[1] Average number of nucleotides compared

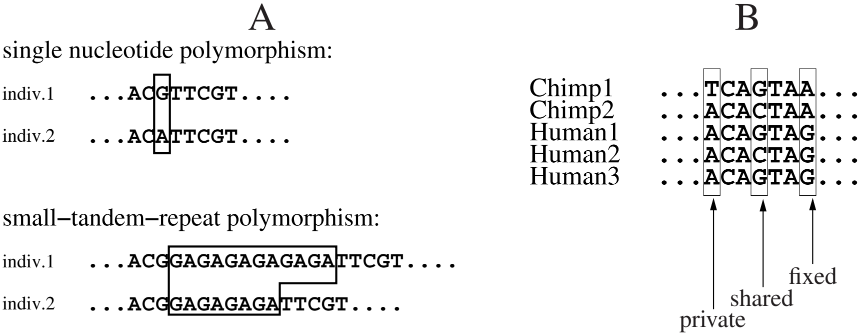

A
single nucleotide polymorphism:
indiv.1 ...ACGTTCGT....
indiv.2 ...ACATTCGT....
small–tandem–repeat polymorphism:
indiv.1 ...ACGGAGAGAGAGAGATTCGT....
indiv.2 ...ACGGAGAGAGATTCGT....
B
Chimp1 ...TCAGTAA...
Chimp2 ...ACACTAA...
Human1 ...ACAGTAG...
Human2 ...ACACTAG...
Human3 ...ACAGTAG...
fixed
shared
private

Fig. 10.3. A: Comparisons within a species or a population. The sketch shows the difference between single nucleotide *vs.* length polymorphisms; the lower example depicts a dinucleotide tandem repeat as found in microsatellites. The two alleles differ in their numbers of dinucleotide repeats. **B**: Comparison within and between species or populations. According to their distribution, the polymorphisms found in the five individuals are shared, private, or fixed differences.

quences of a particular gene from chimpanzee with three sequences of the same gene from human (Fig. 10.3). A *private* polymorphism is found only within one of the two species. In our example the private polymorphism only occurs in the chimpanzee. In contrast, a *shared* polymorphism is variable in both species. A *fixed difference*, finally, is monomorphic within a species but polymorphic between (Fig. 10.3B).

10.2 The Neutral Theory

The contribution of natural selection to shaping the genetic diversity around us has been debated for over a century. Darwin himself wrote [44, p. 103]

> I am inclined to suspect that we see in these polymorphic genera variations in points of structure which are of no service or disservice to the species.

In the 1920s the American population geneticist Sewall Wright and his English colleague Ronald A. Fisher started a controversy over the relative importance of adaptive and non-adaptive evolution that was to last until Fisher's death in 1962 [204]. Fisher saw little role for non-adaptive evolution, while this was an important component of Wright's models of evolution. However, by the 1950s Fisher's view that essentially all heritable differences were adaptive had gained an ascendancy that is rather surprising, given Darwin's skepticism in this matter.

In 1966, two studies on genetic diversity in natural populations were published that challenged the prevailing view. One study focused on genetic diversity in the fruit fly *Drosophila melanogaster* [118, 167], the other on human genetic diversity [104] and both were based on allozyme data. In order to collect these, total protein is extracted from an organism, run on a non-denaturing gel, and then stained for specific enzyme activity. The result is one or more bands on the gel, whose positions indicate the allele of the gene encoding the enzyme [210]. In *Drosophila*, 39% of the loci investigated were polymorphic, leading to the estimate that 8–15% of all loci in an individual fruit fly were heterozygous [167]. In humans, the amount of genetic diversity uncovered was also surprisingly high [104]. This lead Kimura [139] and, independently, King and Jukes [145] to propose that the unexpectedly high levels of polymorphism at the molecular level were best explained by assuming that the vast majority of them had no influence on an organism's fitness. Under this neutral hypothesis of molecular evolution, changes in allele frequencies between generations are due to chance, that is genetic drift, alone and not due to selection. As we will see in more detail below, the main effects of drift are:

1. Undirected change of allele frequencies.
2. Removal of established genetic diversity. The speed with which this happens is inversely proportional to population size. Loss of diversity through drift is therefore mainly relevant in small populations.
3. Removal of new mutations. This is important both in small as well as in large populations because mutations are the only source of genetic diversity and hence constitute the "raw material" of evolution.

The birth of new alleles through mutation and their death through drift tend to come to a mutation drift equilibrium. According to the neutral theory, most genetic diversity, which can be seen today, is a reflection of this equilibrium [141].

Two lines of reasoning were used to support the neutral theory. The first was put forward earlier by Haldane [102] who proposed that allele substitution is the result of positive selection acting on favorable alleles. However, each substitution is inevitably linked to a number of genetic deaths of those individuals that do not carry the favorable allele. This effect was called the *genetic load* or the *cost of natural selection*. Haldane [102] had estimated that a species could tolerate at most one substitution every 300 generations in order to cope with the *cost of natural selection*. Kimura examined globins (see Table 9.1) and other protein sequences from various vertebrates and calculated an average rate of 2.8 amino acid substitution per 10^7 years per 100 amino acids [139]. Assuming a size of the mammalian genome of $4 \cdot 10^9$ nucleotides, he estimated that this rate would correspond to 0.57 nucleotide replacements per year, or one substitution per 1.75 years. For the average mammalian generation size of four years, this amounts to 2.3 substitutions per generation [139]. This figure contrasts strongly with Haldane's result, unless the assumption that all substitutions are selective is dropped.

The second line of reasoning, put forward by King and Jukes in 1969, stressed the fact that the genetic code allowed for synonymous substitutions which were not affecting the encoded protein and were therefore invisible to selection (Table C.4). The discovery only a few years earlier of the genetic code and its degeneracy [191] was an essential prerequisite to this argument.

In this chapter we consider neutral models of evolution that move forward in time, while backward in time models are introduced in Chapter 11. We start by explaining how population genetic quantities are simulated forward in time, before approaching evolutionary models of increasing complexity and hence realism.

10.3 Modeling Evolution Forward in Time

Evolutionary processes tend to unfold over long periods of time. There are exceptions to this rule, such as the evolution of pathogenic viruses and bacteria, which is quick enough to generate vaccine-resistant or antibiotic-resistant strains within a few years—a very short period on an evolutionary time scale. However, even with microbes it takes dedication to observe evolution directly [61].

A popular substitute for direct observation of complex dynamical processes are computer simulations. Figure 10.4 demonstrates how simple simulations of genetic quantities can be carried out forward in time: the population, usually consisting of a large set of haplotypes or genotypes, is represented in its entirety and reproduces from one generation to the next by resampling with replacement. Genetic quantities of interest, for example the frequency of an allele, are then computed either from the entire population, or from a random sample.

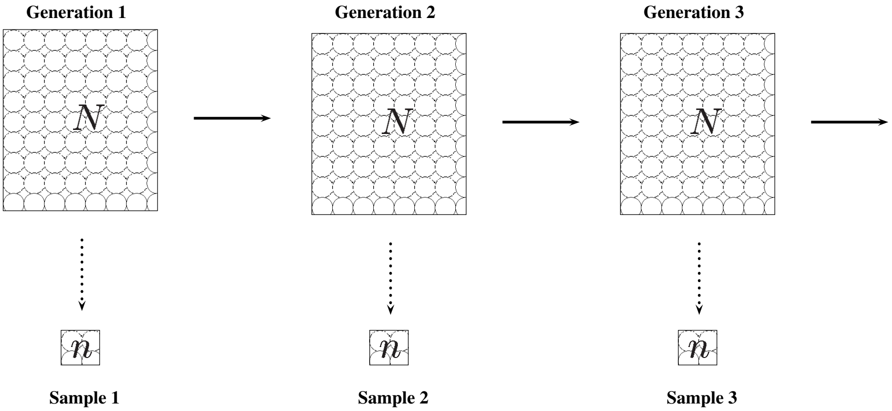


Fig. 10.4. Simulating the evolution of a population (size N) on a computer by moving forward in time. As the simulation moves from one generation to the next, samples (size n) are drawn from which the statistic of interest, e.g. the frequency of a particular allele, is calculated.

Consider one generation of a population of size $N = 6$ consisting of two alleles, A and a: $g_1 = \{a, a, A, A, A, A\}$. The frequency of allele a, p_{a}, is equal to $1/3$ and $p_{\mathrm{A}} = 1 - p_{\mathrm{a}} = 2/3$. Now simulate the process of evolving from the present generation to the next by rolling a die six times returning, say, $1, 3, 6, 4, 3, 6$. For the 1 we draw the first allele (a), for the 3 the third (A), and so on. This leads to the allele configuration $g_2 = \{a, A, A, A, A, A\}$ in the second generation. The allele frequencies have changed and we repeat the random drawing of alleles to produce the next generation. This time the result is $g_3 = \{A, A, A, A, A, A\}$; in other words, A has become fixed and a extinct, as in the absence of mutation there is no way of regenerating a. Thus, genetic diversity has been lost.

A standard measure of genetic diversity is the *heterozygosity*, H, which is the probability that two randomly drawn alleles are different. So for g_1 we have

$$H(g_1) = \frac{\binom{2}{1}\binom{4}{1}}{\binom{6}{2}} = 0.533$$

and for g_2 the diversity is

$$H(g_2) = \frac{\binom{1}{1}\binom{5}{1}}{\binom{6}{2}} = 0.333.$$

In g_3 finally, $H(g_3)$ drops to 0 and remains stuck at this value in all subsequent generations. Our simple experiment reconfirms two important points about drift: (i) it changes allele frequencies and (ii) it removes alleles from the population irreversibly.

10.4 The Neutral Wright-Fisher Model

The neutral *Wright-Fisher model* is the simplest population genetic model and makes the following assumptions. A population is represented as a set of genes. When considering diploid organisms, i.e. those with a double set of chromosomes such as humans, there are $2N$ genes in a population of N organisms. The model further assumes that population size is finite and remains constant over time. Generations are discrete and non-overlapping; therefore, they can be indexed with integers t, $t+1$ and so forth. The genes of generation $t+1$ are drawn randomly from the *gene pool*, i.e. the genes present in generation t (Fig. 10.5). Drawing of genes is carried out with replacement since any particular gene may be passed on to more than one offspring. The biological counterpart to random sampling is the assumption of *random mating* or *panmixis* in sexually reproducing organisms. Randomly mating individuals have no preference for a specific partner based on phenotypic traits or geographic proximity.

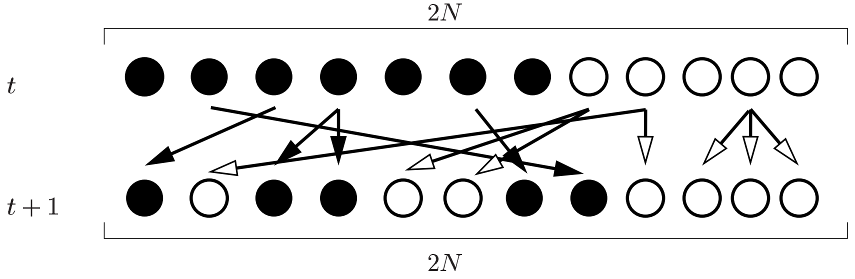


Fig. 10.5. Evolution in a Wright-Fisher population. Arrows indicate the transition from generation t to $t+1$ by drawing with replacement $2N$ genes from generation t. Note that the frequency of alleles (represented as filled and empty circles) may change due to genetic drift. In this example, the frequency of black alleles decreases from $p(t) = 7/12$ to $p(t+1) = 5/12$.

10.4.1 Fixation and Loss of Alleles

To formalize the ideas just presented, assume there are two versions of a given gene, the alleles A and a. Let them be present in a finite diploid population of size N in generation t with relative frequencies

$$p_A(t) = p(t) = \frac{i}{2N}$$

and

$$p_a(t) = 1 - p(t) = \frac{2N - i}{2N},$$

where i is a number between 0 and $2N$. The frequency $p(t+1)$ depends on the outcome of a binomially distributed random variable with parameters $2N$ and $p(t)$. The possible states $j = 0, \ldots, 2N$ in generation $t+1$ have therefore the conditional probabilities

$$\text{Prob}(j, t+1 | i, t) = \binom{2N}{j} \left(\frac{i}{2N}\right)^j \left(1 - \frac{i}{2N}\right)^{2N-j}. \qquad (10.1)$$

The righthand side of Equation (10.1) is independent of t. This property is called *homogeneity*. The associated stochastic process $(X(t))_t$ is Markovian with stationary transition probabilities and transition matrix

$$\mathcal{P} = \left(\binom{2N}{j} \left(\frac{i}{2N}\right)^j \left(1 - \frac{i}{2N}\right)^{2N-j}\right)_{ij} \qquad i, j \in 0, \ldots, 2N.$$

Furthermore, the probabilities $\text{Prob}(j, t+u | i, t)$ are obtained from the matrix product $\mathcal{P}^u$, for any t and u. The boundaries $j = 0$ and $j = 2N$ are *absorbing* states. This property corresponds to our observation made in the die rolling experiment that once an allele is lost, it cannot be restored. The complementary allele is *fixed* and remains so forever. Formally, if $p(t^*) = 1$ for some time t^* then $p(t) = 1$ for all $t > t^*$. Two realizations of this process are depicted in Figure 10.6.

Claim 10.1 *Either one of the two alleles will eventually be fixed, i.e. there is a time* t^**, such that*

$$\textit{Prob}(X(t) = 0 \vee X(t) = 2N) = 1 \qquad (10.2)$$

for all times $t > t^*$.

PROOF. Let $X(t)$ be the number of A-alleles at time t and $B(n, p)(x)$ the cumulative binomial probability with parameters n and p, i.e.

$$B(n, p)(x) = \sum_{y=0}^{x} \binom{n}{y} p^y (1-p)^{n-y}.$$

For any time t and independently of any initial conditions it holds that

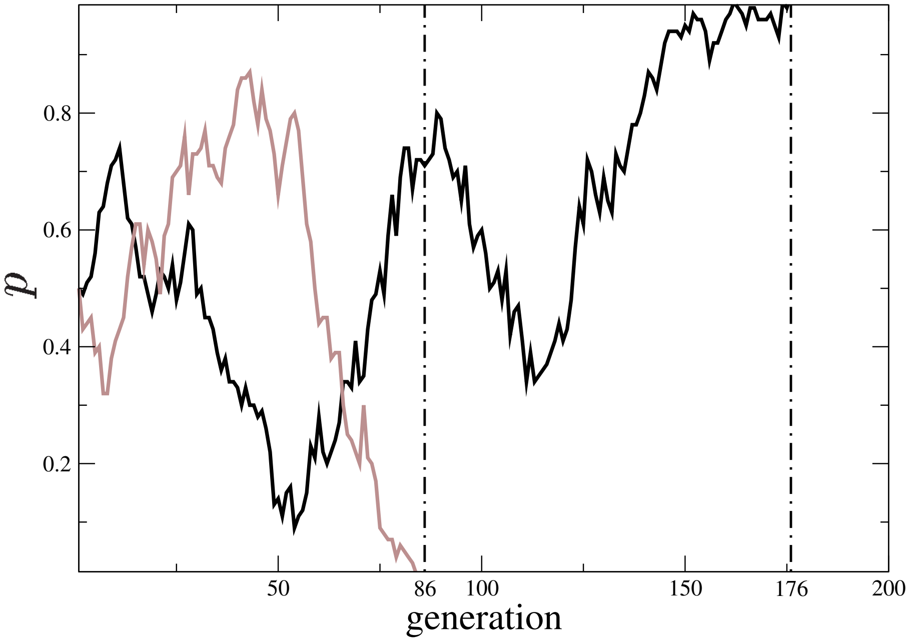


Fig. 10.6. Simulation of genetic drift over 200 generations in a two-allele system. In the gray trajectory allele A is lost at time $t = 86$, in the black trajectory allele A is fixed at $t = 176$. The initial frequency of allele A in both cases is $p(0) = 0.5$ and population size is $N = 100$.

$$\begin{aligned}\text{Prob}(0 < X(t+1) < 2N|p(t)) &= B(2N,p(t))(2N-1) - B(2N,p(t))(0)\\ &= 1 - p(t)^{2N} - (1-p(t))^{2N}\\ &\leq 1 - \left(\frac{1}{2N}\right)^{2N}.\end{aligned}$$

The last inequality is valid for any value of $p(t)$. Thus, the probability

$$\text{Prob}\,(0 < X(\tau) < 2N, \text{for all } \tau \leq t) \leq \left(1 - \left(\frac{1}{2N}\right)^{2N}\right)^{t}. \tag{10.3}$$

The limit of the right side of Equation (10.3) is 0 as t grows and N remains constant. The probability that X does not hit one of the boundaries before time t can be made arbitrarily small if t is allowed to be sufficiently large. □

Loss or fixation of an allele by drift is certain, but in large populations either outcome may take very long to reach. For large N the expression $(2N)^{-2N}$ is close to 0, thus the righthand side of Equation (10.3) is close to one. In the limit of infinitely large populations, there is no drift and allele frequencies stay constant in time. This is one of the two assertions of the Hardy-Weinberg law.

10.4.2 The Hardy-Weinberg Law

Consider again two alleles, A and a. Let their frequencies in an infinitely large, panmictic population be denoted by p_A and p_a. The three possible genotypes, the two homozygotes *AA* and *aa* and the heterozygote *Aa*, have frequencies p_{AA}, p_{Aa}, and p_{aa}. The Hardy-Weinberg law states that (i) from the second generation onwards the allele frequencies remain constant forever and that (ii) the genotype frequencies are uniquely determined by the allele frequencies and *vice versa* by the relationship

$$\begin{aligned} p_{AA} &= p_A^2 \\ p_{Aa} &= 2p_A p_a \\ p_{aa} &= p_a^2 . \end{aligned}$$

In other words, alleles are assembled independently into genotypes. The law is named after George Hardy and Wilhelm Weinberg, who formulated it independently in 1908. In practice, one often finds the Hardy-Weinberg law to be violated. This is mainly due to the fact that natural populations are not infinitely large. Other possible factors that may lead to deviations from the Hardy-Weinberg frequencies are differential action of natural selection upon different genotypes, non-random mating between individuals, or population substructure instead of panmixis.

10.4.3 Fixation Probability and Time to Fixation

We return now to a population of finite size N. In contrast to Claim 10.1, stating that either of two alleles will eventually be fixed, the following statement concerns the fixation probability of a particular allele.

Claim 10.2 *The fixation probability of an allele equals its current frequency* $p(t_0)$. *Formally,*

$$P_{\text{fix}} = \text{Prob}(X(t^*) = 2N \text{ for some } t^* > t_0 | p(t_0)) = p(t_0) . \tag{10.4}$$

PROOF. There are several proofs of this claim. Recurrence theory of finite state Markov chains provides the adequate framework and a detailed treatment of this can be found in a monograph by Ewens [65]. Here, we only note the following. As a consequence of Claim 10.1

$$P_{\text{fix}} = \text{Prob}(\lim_{t\to\infty} X(t) = 2N | p(t_0)) .$$

Furthermore, the limiting matrix $\mathcal{P}^\infty = \lim_{u\to\infty} \mathcal{P}^u$ has the form

$$\mathcal{P}^\infty = \begin{pmatrix} 1 & 0 \cdots 0 & 0 \\ \frac{2N-1}{2N} & 0 \cdots 0 & \frac{1}{2N} \\ \vdots & \vdots \; \vdots \; \vdots & \vdots \\ \frac{1}{2N} & 0 \cdots 0 & \frac{2N-1}{2N} \\ 0 & 0 \cdots 0 & 1 \end{pmatrix} .$$

Given that $p(t_0) = i/(2N)$ for some i, the initial distribution in the state space $\{0, 1, ..., 2N\}$ is $(0, ..., 0, 1, 0, ..., 0) = \pi_i$, where entry 1 is the i-th entry in this vector. By evaluating the product $\pi_i \cdot \mathcal{P}^\infty$ at its $2N$-th entry, one obtains

$$(\pi_i \cdot \mathcal{P}^\infty)_{2N} = \frac{i}{2N}\,.$$

□

In fact, this is true for any time t: given the process is in state i at time t, the fixation probability for allele A is equal to

$$\frac{i}{2N}.$$

The reason is, once again, the time-homogeneity of the underlying Markov process.

How long does it take for an allele to be fixed? To answer this question, the mean absorption time $\bar{t}_{\text{absorb}}(i)$ for, say, allele A with current frequency $p(t_0) = i/(2N)$, has to be determined. Note, that the conditional mean absorption time $\bar{t}_{\text{absorb}}(i)$ is different from the conditional mean fixation time, $\bar{t}_{\text{fix}}(i)$. For the latter, the additional condition that A will be fixed (and not lost) is imposed (Fig. 10.7). A rigorous treatment of this problem either within the theory of discrete Markov chains or con-

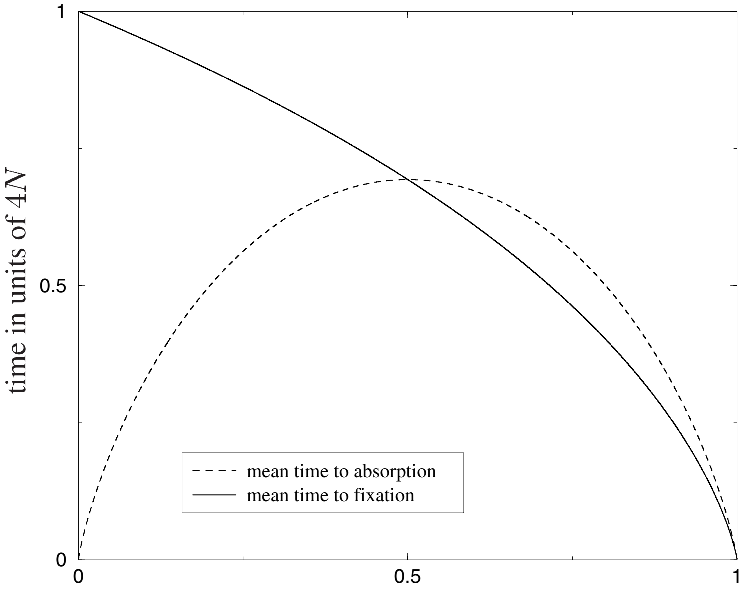


Fig. 10.7. Time to absorption and time to fixation of an allele as a function of its current frequency. Time is measured in units of $4N$ generations.

tinuous diffusion processes can be found in the book by Ewens [65]. However, there is a very instructive approximate solution to this problem. Let $x = i/(2N)$ and $x + \delta x = j/(2N)$. From Equation (10.1) one obtains for the expected change in allele frequency

$$E(\delta x) = E(\frac{j-i}{2N}) = \frac{1}{2N}\left(\sum_{k=0}^{2N} k\,\mathrm{Prob}(k,t+\delta t|i,t)\right) - \frac{i}{2N} = 0$$

and, similarly, for the second moment

$$E(\delta x)^2 = \frac{1}{4N^2}E((j-i)^2) = ... = \frac{x(1-x)}{2N}\,.$$

Treating allele frequencies as a continuous variable x, we write $t(x)$ instead of $t(i)$. Mean absorption time is $\bar{t}_{\text{absorb}}(x)$, given the process is currently in state x. At the next point in time, the process is expected to be in some state $E(x + \delta x)$ and the counter of the time units which the process has spent before reaching $x = 0$ or $x = 1$ is increased by one. Formally,

$$t(x) = 1 + E(t(x + \delta x))\,.$$

Assuming that $t(x)$ is twice differentiable we approximate the above equation by its Taylor series up to order two and obtain

$$\begin{aligned} t(x) &\approx 1 + E(\delta x)\left(t(x)' + \frac{1}{2}E(\delta x)^2(t(x))''\right) \\ &= 1 + t(x) + \frac{x(1-x)}{4N}(t(x))''\,. \end{aligned}$$

Thus, one has to solve

$$x(1-x)(t(x))'' = -4N \tag{10.5}$$

subject to the boundary conditions $t(0) = t(1) = 0$. The solution is

$$t(x) = -4N\,(x\log(x) + (1-x)\log(1-x))\,. \tag{10.6}$$

With the above choice of boundary conditions, Equation (10.6) represents the mean time to absorption, $\bar{t}_{\text{absorb}}(x)$, given the current allele frequency is x. In particular, for a newly arising allele with frequency $x = 1/(2N)$, the mean absorption time is

$$\bar{t}_{\text{absorb}}(1/(2N)) \approx 2 + 2\log(2N)\,.$$

For the mean conditional fixation time $\bar{t}_{\text{fix}}(x)$ essentially the same arguments can be invoked. One only has to note that the transition probabilities p_{ij} (Eq. (10.1)) have to be replaced by the conditional transition probabilities p^*_{ij} that allele A will not be lost. At every generation, it has to be ensured that allele A leaves at least one offspring. Therefore, only $2N - 1$ alleles can be drawn at random and p^*_{ij} is the probability that among $2N-1$ there are exactly $j-1$ alleles of type A. Therefore, these transition probabilities are

$$p^*_{ij} = \binom{2N-1}{j-1}\left(\frac{i}{2N}\right)^{j-1}\left(1-\frac{i}{2N}\right)^{2N-j}. \tag{10.7}$$

The differential equation, analogous to Equation (10.5), now reads

$$(1-x)(t(x))' + \frac{1}{2}x(1-x)(t(x))'' = -2N\,.$$

Its solution, subject to the boundary condition $t(1)=0$, is

$$t(x) = -4N\left(\frac{1-x}{x}\log(1-x)\right). \tag{10.8}$$

In particular, for a newly arising allele with frequency $x = 1/(2N)$, one obtains

$$\bar{t}_{\text{fix}}(1/(2N)) \approx 4N\,. \tag{10.9}$$

Finally we mention the average conditional time to loss of an allele; it is

$$\bar{t}_{\text{loss}}(x) = -\frac{4Nx}{1-x}\log(x)\,. \tag{10.10}$$

Given that the allele is lost and has frequency $x = 1/(2N)$, it takes only

$$\bar{t}_{\text{loss}}(1/(2N)) \approx 2\log(2N)$$

generations on average until it is lost.

An exact formulation of these arguments was provided by Kimura and Ohta [143] based on the theory of diffusion processes.

10.4.4 Loss of Genetic Diversity

We now return to the study of genetic diversity under a two-allele model. Allele A is sampled with probability p, allele a with probability $1-p$. There are two ways in which two different alleles may be sampled: choose allele A first and then allele a or *vice versa*. Therefore, genetic diversity in the two-allele model is

$$H(t) = 2p(t)(1-p(t))\,. \tag{10.11}$$

Note that $H(t)$ can also be interpreted as the expected *heterozygosity* (hence the H) of a diploid individual, that is, the probability that a diploid individual carries two different alleles. Let $X(t)$ be a binomially distributed random variable with parameters $2N$ and

$$p(t-1) = \frac{i}{2N}$$

and note that the second moment may be written in terms of the variance and the first moment as

$$E(X^2) = V(X) + (E(X))^2 \,.$$

The dynamics of H under random drift can then be determined as follows:

$$\begin{aligned}
H(t) &= 2p(t)(1 - p(t)) \\
&= 2E\left(\frac{X(t)}{2N}\left(1 - \frac{X(t)}{2N}\right)\right) \\
&= 2\left(\frac{E(X(t)}{2N} - \frac{E(X^2(t)}{(2N)^2}\right) \\
&= 2\left(\frac{i}{2N} - \frac{1}{(2N)^2}\left(2N\left(\frac{i}{2N}\left(1 - \frac{i}{2N}\right)\right) + \left(2N\frac{i}{2N}\right)^2\right)\right) \\
&= 2\left(p(t-1) - \frac{1}{2N}(p(t-1)(1 - p(t-1))\right) \\
&= 2\left(\left(1 - \frac{1}{2N}\right)(p(t-1)(1 - p(t-1))\right) \\
&= H(t-1)\left(1 - \frac{1}{2N}\right) .
\end{aligned}$$

Iterating the above procedure, one finds

$$H(t) = H(0)\left(1 - \frac{1}{2N}\right)^t \approx H(0)\exp\left(-\frac{t}{2N}\right) . \tag{10.12}$$

Thus, heterozygosity decays under random genetic drift at a rate of $1/(2N)$ per generation. This is another way of stating that drift reduces genetic diversity. As an immediate consequence of Equation (10.12), one finds the "half life" of heterozygosity

$$t_{h/2} = -\frac{\log(2)}{\log(1 - 1/(2N))} . \tag{10.13}$$

With the approximation $\log(1 - 1/(2N)) \approx -1/(2N)$, it follows that

$$t_{h/2} \approx 2N\ln(2) \,. \tag{10.14}$$

The half life of heterozygosity depends linearly on the population size. The above arguments all rest upon the assumption that there are no new mutations entering the population. But in reality, there is continuous influx of new alleles into a population, created by mutation. The key arguments of the neutral theory of molecular evolution rest upon the interplay of drift and mutation, which we consider next.

10.5 Adding Mutation to the Model

Depending on the degree of resolution with which molecular evolution is studied, different models of the mutation process are employed. Here, we describe the finite alleles model, the infinite alleles model, and the infinite sites model.

10.5.1 Finite Alleles Model

The finite alleles model, in particular the two-allele model, has originally been used to describe the evolutionary dynamics in situations where the alleles represent macroscopically observable phenotypes, such as eye or body color. The mutation dynamics in this case is modeled as switching between a given finite number of states. In particular, mutations can also be reversible and there is no influx of new alleles.

In fact, one-locus two-allele models with back mutation belong to the classical repertoire of theoretical population genetics and have been treated by all three founders of the subject, Haldane, Wright, and Fisher. In this case there exist nontrivial equilibria of the allele frequencies. In the presence of drift, there is a stationary density of the allele frequency distribution [262]: given a diploid population of size N, two alleles A_1 and A_2, and mutation rates μ_1 (for mutation from A_1 to A_2) and μ_2 (for mutation from A_2 to A_1) per chromosome per generation, the stationary distribution of the frequency x of allele A_1 is

$$f(x) = \frac{\Gamma(4N\mu_1 + 4N\mu_2)}{\Gamma(4N\mu_1)\Gamma(4N\mu_2)} x^{4N\mu_2 - 1}(1-x)^{4N\mu_1 - 1} .$$

The mean and variance are

$$E(f(x)) = \frac{\mu_2}{\mu_1 + \mu_2}$$

and

$$V(f(x)) = \frac{\mu_1\mu_2}{(\mu_1 + \mu_2)^2(4N\mu_1 + 4N\mu_2 + 1)} .$$

If $\mu_1 = \mu_2 = \mu$, then

$$f(x) = \frac{\Gamma(8N\mu)}{(\Gamma(4N\mu))^2} x^{4N\mu - 1}(1-x)^{4N\mu - 1} .$$

Abbreviating $4N\mu$ by θ, the probability that two chromosomes, drawn at random from the population, are of the same allelic type, is

$$\int_0^1 (x^2 + (1-x)^2) df(x) = \frac{1+\theta}{1+2\theta} .$$

This expression can also be interpreted as the expected fraction of homozygotes in the population. The expected fraction of heterozygotes is then

$$E(H) = \frac{\theta}{1+2\theta} .$$

With the advent of protein electrophoresis in the 1960s and the observation of unexpectedly high diversity within species [167, 165], the finite alleles model started to be superseded by the infinite alleles model.

10.5.2 Infinite Alleles Model

With the infinite alleles model [142] it is possible to account for continuous mutational influx into a population. In fact, its name derives from the assumption that any mutation event creates a new allele, which was previously not seen in the population. There is no back mutation to previous states. As a consequence, two alleles can be identical only due to shared ancestry rather than chance mutation to the same allele. Hence identical alleles are said to be *identical by descent*. In the framework of the infinite alleles model it is possible to decide whether any two alleles are identical or not. However, it is not possible to quantify the difference, for instance in terms of the number of nucleotide mismatches between two different alleles. Put more formally, allele space is a metric space with the trivial metric

$$d(A_i, A_j) = \begin{cases} 0, & \text{if } A_i = A_j \\ 1, & \text{if } A_i \neq A_j. \end{cases}$$

The infinite alleles model is reasonable under many real-world scenarios, when the level of resolution is fine enough. As an example, consider again the data from Kreitman shown in Table 10.1. While he found only two electrophoretic variants in his sample of eleven genes, the number of alleles which are distinguishable at the level of nucleotides was nine. To convince yourself why back mutation can be neglected when dealing with (infinitely) many alleles, consider a sequence of 180 nucleotides. Once this sequence has started mutating, it is unlikely to ever return to its original state, as there are $4^{180} \approx 2 \cdot 10^{108}$ distinct sequences it can reach. Compare this to the estimate that our universe has a volume of 10^{108} Å^3 [59, p.35] and you can appreciate that the space of possible nucleotide sequences is truly vast even for sequences of only moderate length.

10.5.3 Infinite Sites Model

Under the infinite sites model each mutation affects a different position along a stretch of DNA that has never mutated before. This model is realistic when describing and interpreting the evolution of sets of DNA sequences, where the mutation rate per site is low. Allele space in the infinite sites model is equipped with a non-trivial metric. The natural distance between any two sequences is the number of sites at which the two aligned sequences differ. If all sequences have the same length and are alignable without gaps, this metric is also called Hamming distance.

10.6 Mutation Drift Balance

10.6.1 The Rate of Fixation

How fast and how often new alleles are fixed is determined by the time to fixation and the rate of fixation. Figure 10.8 shows sample trajectories of newly arising alleles which eventually reach fixation. Under the infinite alleles or infinite sites models

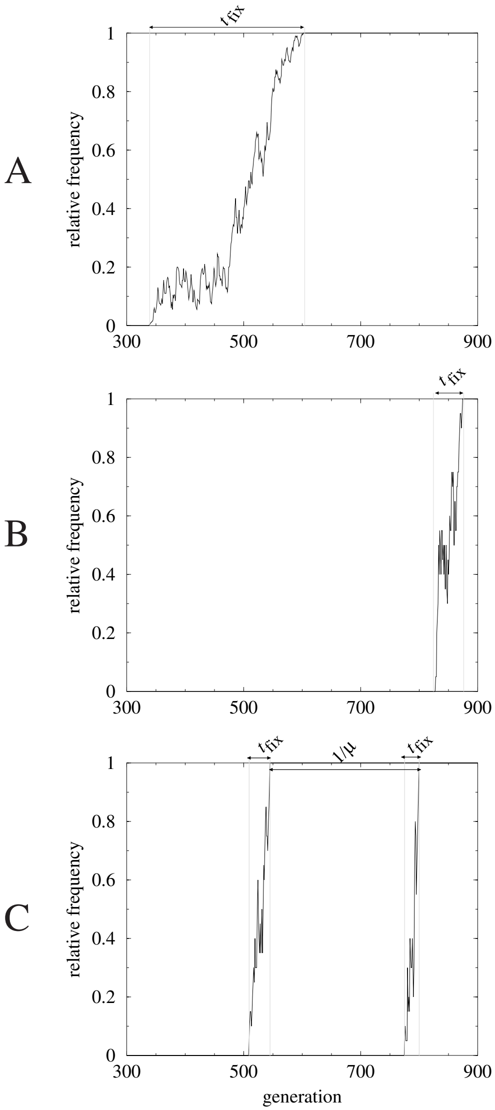


Fig. 10.8. Fixation of new alleles under an infinite alleles or infinite sites model. X-axis: time in generations, Y-axis: relative frequency of mutant alleles. Shown are only the trajectories of those alleles which eventually reach fixation. **A**: $N = 100$, mutation rate $\mu = 10^{-3}$; **B**: $N = 10$, mutation rate $\mu = 10^{-3}$; **C**: $N = 10$, $\mu = 10^{-2}$. The time to fixation (t_{fix}) depends on N (compare **A** and **B**), while the rate of fixation depends on μ (compare **B** and **C**). The average time to fixation is $t_{\text{fix}} = 4N$ generations. At any given time a random number κ of alleles is present in the population. This number depends on N and μ. Its lower bound estimate is $\underline{\kappa} = 1 + 4N\mu$ (Eq (10.20)). For our examples $\kappa = 1.4$ (**A**), 1.04 (**B**) and 1.4 (**C**). Note that the figures do not display the trajectories that do not reach fixation, which are nevertheless included in this estimate.

and in a diploid population, $2N\mu$ new alleles are generated independently in each generation. Any individual allele is eventually either lost or fixed. There is no equilibrium frequency of individual alleles different from 0 or 1. The rate of fixation is the product of newly generated mutants per generation times their probability of fixation,

$$2N\mu \cdot \frac{1}{2N} = \mu.$$

This is the rate of molecular evolution, or rate of substitution, introduced in Chapter 9.

The equality of substitution and mutation rates under the neutral infinite alleles model is an immediate, but nevertheless surprising result given our intuition that the mutation rate is some sort of universal constant, while the rate of substitution should be a function of the population size. Both intuitions are correct and under neutrality the corresponding terms cancel out. This calculation does not hold if mutations are non-neutral.

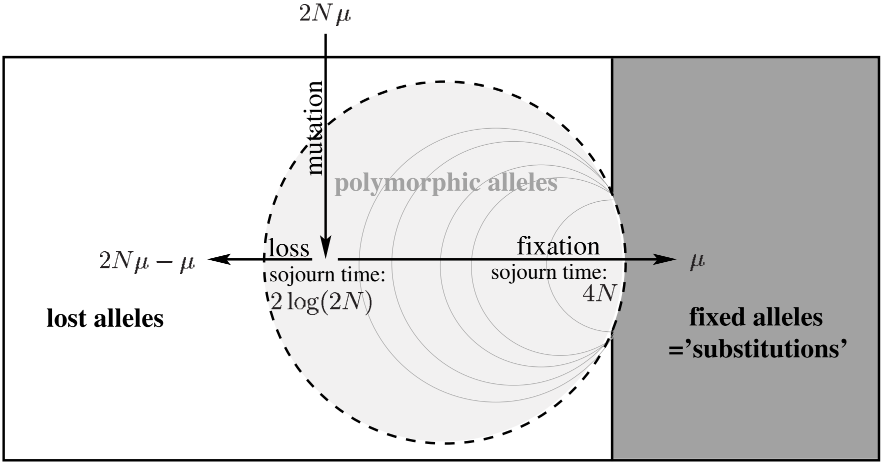


Fig. 10.9. Mutation drift balance may be visualized as a balance of influx and outflux of alleles. The mean sojourn time of a new allele destined to be lost is $2\log(2N)$ (Eq. (10.10)). The mean time to fixation is $4N$ (Eq. (10.8)).

10.6.2 Number of Alleles

We have seen that each individual allele is either lost or goes to fixation. However, as genetic diversity is lost through drift, new diversity is generated through mutation and these two factors lead to an equilibrium between the influx of mutations into the gene pool and their subsequent outflux (Fig. 10.9). We can now ask, is there an equilibrium under mutation and drift for the number of alleles which are simultaneously present in the population at any given point in time? A lower bound estimate of this number was derived by Kimura and Crow [142]. Consider two randomly chosen alleles in generation t. Let $F(t)$ be the probability that they are identical by descent, i.e. derived from either the same parent or from two alleles which were already identical by descent in generation $t-1$. The quantity F is also called *homozygosity*. The probability that two alleles are identical by descent in generation t is derived as follows. Either they have descended from the same allele in generation $t-1$. The probability of this is $p = 1/(2N)$ in a diploid population of N individuals; or they have not descended from the same allele in generation $t-1$ ($q = 1 - 1/(2N)$), but have descended from the same allele in some previous generation; by definition this is $F(t-1)$. In both cases neither of the two alleles is allowed to mutate while being passed from generation $t-1$ to generation t. The probability of this is $(1-\mu)^2$. Summarizing, we can write

$$F(t) = (1-\mu)^2 \left(\frac{1}{2N} + \left(1 - \frac{1}{2N}\right) F(t-1) \right). \tag{10.15}$$

In order to solve Equation (10.15) for $F(t)$, we simplify the algebra by remembering that biologically reasonable mutation rates are small and populations large. Hence, we ignore terms multiplied by μ^2 and μ/N and write

$$F(t) \approx \frac{1}{2N} + \left(1 - \frac{1}{2N}\right) F(t-1) - 2\mu F(t-1). \tag{10.16}$$

In an equilibrated population the fraction of alleles which are identical by descent remains constant between generations, i.e. $F(t-1) = F(t) = \bar{F}$. Thus, when replacing $F(t-1)$ and $F(t)$ in Equation (10.16) by $\bar{F}$, equilibrium homozygosity can be written as

$$\bar{F} = \frac{1}{1+4N\mu}. \tag{10.17}$$

In case the number of alleles and their frequencies are known, homozygosity is calculated as

$$F = \sum_{i=1}^{\kappa} x_i^2 , \tag{10.18}$$

where κ is the number of alleles and x_i is the frequency of allele i. The proportion of homozygotes (F) in the population is minimal if all alleles are equally frequent,

$$x_i = \frac{1}{\kappa}$$

for all i. In this situation, one has

$$F = \frac{1}{\kappa} = \frac{1}{1+4N\mu} \tag{10.19}$$

and a lower bound estimate for the number of alleles in an equilibrated population therefore is

$$\underline{\kappa} = 1 + 4N\mu \,. \tag{10.20}$$

This number has been called the *effective number of alleles* by Kimura and Crow [142] and numerical values for this are shown in Figure 10.8. Note that the scaled mutation rate, $\theta = 4N\mu$, which plays a central role in population genetics theory, is featuring again in this formula. For non-uniformly distributed allele frequencies the number of alleles is larger than $\underline{\kappa}$. In Section 10.7 we will come back to a formula for the number of alleles in a sample drawn from an equilibrated population.

10.6.3 Genetic Diversity

There are several measures of genetic diversity. One of them is the number of different alleles in a population. This statistic has the advantage that it is easy to measure. However, its disadvantage is that it is dependent on the size of the sample from which diversity is estimated. This makes it hard to compare measurements which are based on different samples with different sizes.

Another measure of genetic diversity is the proportion of heterozygotes in a population or, equivalently, the probability with which two randomly chosen alleles are different. In case of known alleles and allele frequencies, heterozygosity is

$$H = 1 - \sum_i x_i^2 \,.$$

On the other hand, and as an immediate consequence of Equation (10.17), equilibrium heterozygosity is

$$\bar{H} = 1 - \bar{F} = \frac{4N\mu}{1 + 4N\mu} = \frac{\theta}{1 + \theta} \,. \tag{10.21}$$

This result entails one of the corner stones of the neutral theory of evolution [139]. Diversity in neutrally evolving populations is monotonically increasing with the scaled mutation rate θ. In fact, if $\theta \ll 1$, then $H \approx \theta$. Equation (10.21) applies to average heterozygosity. Under the action of mutation and drift, heterozygosity is a Markov process with stationary transition probabilities. For each point in time, heterozygosity is a random variable, $H(t)$. Three realizations of this stochastic process under the infinite alleles model are shown in Figure 10.10. It demonstrates that heterozygosity as a random variable in time fluctuates around the value calculated in Equation (10.21). The entire distribution of H at a single locus and under the infinite alleles model has been derived by Fuerst and colleagues [82]. In particular, for the variance, $V(H)$, they obtained

$$V(H) = \frac{2\theta}{(1 + \theta)^2 (2 + \theta)(3 + \theta)} \,. \tag{10.22}$$

Equation (10.21) plays a central role and will be rederived with different arguments in Chapter 11. We consider further its implications. Notice first that Equation (10.21) establishes a relationship between heterozygosity H, which can be

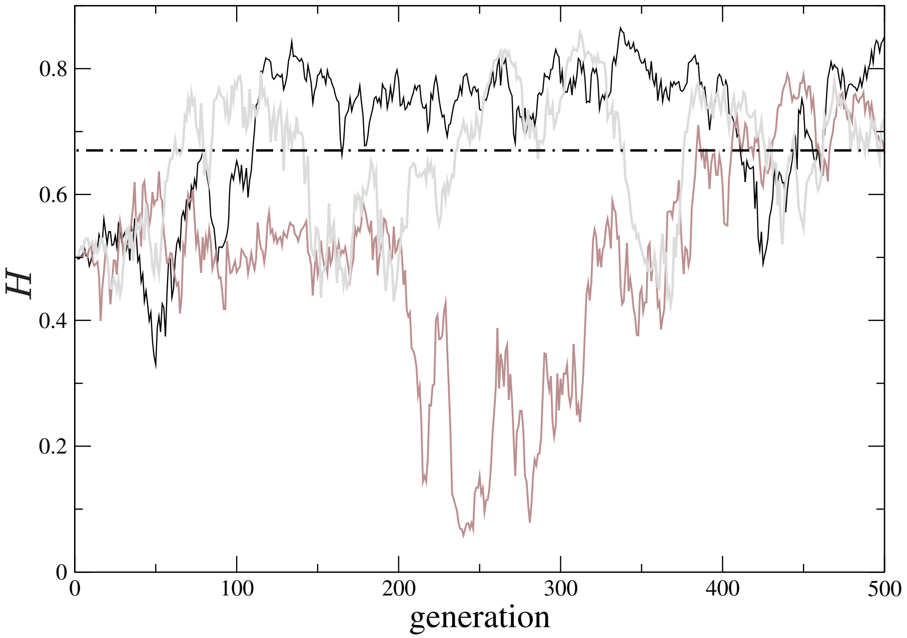


Fig. 10.10. Simulation of heterozygosity at three independent loci evolving under the infinite alleles model. Each trajectory represents the heterozygosity $H(t)$ at one locus. Input parameters are $N = 50$ and $\mu = 0.01$, which implies a mean heterozygosity of $H = \frac{4N\mu}{4N\mu+1} = 0.67$ (dotted line). Notice the fluctuations around the expected heterozygosity. For the parameters considered the standard deviation is 0.15 (Eq. (10.22)).

measured, and the unknown quantity $4N\mu$. As an application consider the data set shown in Table 10.1. If we ignore indels, we observe eight unique alleles and one allele which occurs three times (Fr-F, Wa-F, Af-F). Hence, the observed heterozygosity is $H_{\text{obs}} = (1 - 8 \cdot (1/11)^2 - 1 \cdot (3/11)^2) \cdot 11/10 = 0.95$, where the factor $n/(n-1) = 11/10$ corrects for the fact that we are using sample frequencies rather than population frequencies [121]. Based on Equation (10.21) one would estimate the scaled mutation rate to be $\hat{\theta} = 17.33$, where the hat indicates that the value is an estimate. Second, there are estimates of the (unscaled) mutation rate μ available [51], which means that Equation (10.21) can be used to calculate the population size. For *Drosophila*, Drake and his colleagues reported a mutation rate of $\mu = 8.5 \cdot 10^{-9}$ per bp per generation. Thus, the mutation rate for the 2,721 bp investigated in the *Adh* region would be $2.31 \cdot 10^{-5}$ per generation. This yields an estimate of the population size $\hat{N} \approx 187{,}000$. We need to stress that population size refers here, as always in our book, to the *effective* population size, not the census population size. Roughly, effective population size is the number of reproducing individuals. This number is generally much smaller than the actual number of individuals. Finally, Equation (10.21) predicts that heterozygosity approaches unity with increasing population size. This prediction was central to attempts in the 1970s to falsify the neutral theory by measuring H in the largest populations known, those of bacteria. The first of these studies came to the conclusion that for *Escherichia coli* $H \approx 0.2$, which is far from unity [182]. However, the estimation of N from H is fraught with difficulties. One of these follows from the shape of the graph of H as a function of $4N\mu$ shown in Figure 10.11. For large values of H, N cannot be predicted very accurately as the curve flattens out. Moreover, Equation (10.21) is based on the assumption of free recombination, which implies that all loci evolve independently of each other. This assumption is not valid for asexual organisms such as *E. coli* [222, 175].

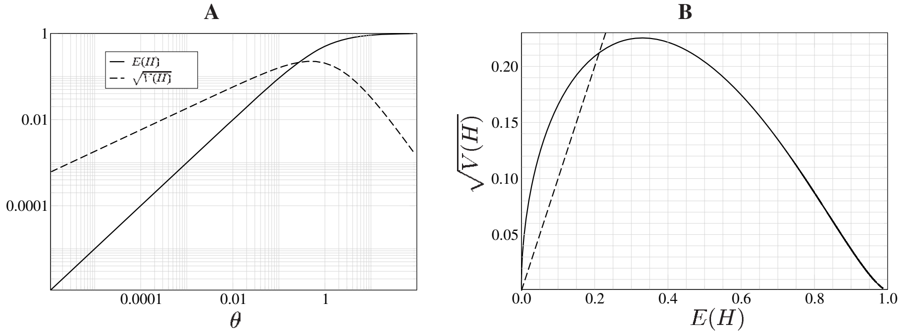


Fig. 10.11. A: Expected heterozygosity $E(H)$ and standard deviation $\sqrt{V(H)}$ as function of the scaled mutation rate $\theta = 4N\mu$ under the neutral infinite alleles model (Eqs. (10.21) and (10.22)). **B:** Standard deviation as function of expected heterozygosity. The dashed line is a plot of $x = y$ and shows that $\sqrt{V(H)} > E(H)$ for small values of $E(H)$.

10.7 Sampling Alleles from Populations

10.7.1 Ewens' Sampling Formula

A cornerstone of the study of the infinite alleles model is Ewens' sampling formula [64]. Consider a population in equilibrium under the neutral infinite alleles model and a sample of size n drawn from the population. The sample configuration is a vector $\mathbf{C}(n) = (C_1(n), ..., C_n(n))$, where the components $C_j(n)$ denote the number of alleles represented exactly j times in the sample. Then, $\sum_{j=1}^{n} jC_j(n) = n$. Ewens' sampling formula gives the probability distribution for the sample configuration. The probability for a particular configuration $\mathbf{c}(n) = (c_1(n), c_2(n), ..., c_n(n))$, where $c_j(n)$ are non-negative integers with the property $n = \sum_{j=1}^{n} jc_j(n)$, is

$$P\left(\mathbf{C}(n) = \mathbf{c}(n)\right) = \frac{n!\theta^{\sum_j c_j(n)}}{\prod_j j^{c_j(n)} \prod_j c_j(n)! \theta_{(n)}}, \tag{10.23}$$

where $\theta_{(n)} = \theta(\theta+1)...(\theta+n-1)$. Let further $\mathcal{K}_n$ be the number of distinct alleles in a sample of n. Then, $\sum_{j=1}^{n} C_j(n) = \mathcal{K}_n$. The distribution of $\mathcal{K}_n$ is

$$P(\mathcal{K}_n = \kappa) = \frac{\mathrm{coeff}(\theta(n), \kappa)\theta^{\kappa}}{\theta_{(n)}} ,$$

where $\mathrm{coeff}(\theta(n), \kappa)$ is the coefficient of θ^{κ} in the expansion of $\theta_{(n)}$.

Ewens' sampling formula is the basis for one possible test of the null hypothesis of neutral evolution. More tests of this hypothesis are discussed in Chapter 12. Here, we apply Equation (10.23) to the experimental data shown in Table 10.1. In this data set sample size is $n = 11$; therefore, we are dealing with a vector $\mathbf{c}(11)$ of the form $\mathbf{c}^*(11) = (8, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0)$. The probability for this configuration is

$$P(\mathbf{C}(11) = \mathbf{c}^*(11)) = \frac{330 \cdot \theta^8}{\theta(11)} .$$

For sample size $n = 11$, there are altogether 56 possible configurations. The number of possible configurations is equal to the number of decompositions of n into integer summands without regard of order [1, sec. 24.2.1]. The probabilities for each configuration depend only on θ. Figure 10.12 shows the probability distribution for different numerical values of θ. The observed configuration is indicated by a vertical black line. Crucial for deciding whether the observation is compatible with the model of neutral evolution is an accurate estimate of θ. One can show that all information

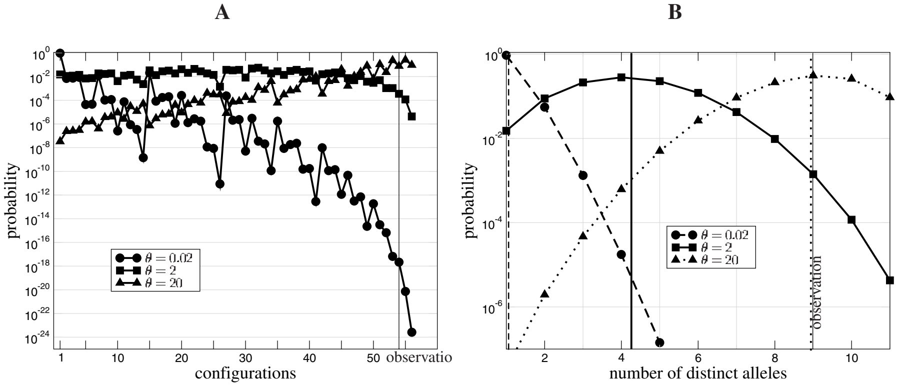


Fig. 10.12. **A**: Probability distribution of the possible configurations for a sample of size $n = 11$ and different values of θ. Configurations are ordered lexicographically. For instance, configuration 1 corresponds to the vector $(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1)$ (i.e. all alleles are identical), configuration 2 to $(0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0)$, and configuration 54 to the observed configuration $(8, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0)$ (i.e. there are eight singletons and one cluster containing three alleles). **B**: Probability distribution for $\mathcal{K}_{11}$ and different values for θ. The observed number of distinct alleles (9) is indicated by the black vertical line. Vertical lines represent the mean values $E(\mathcal{K}_{11})$ for the three choices of θ.

about θ which is deducible from a configuration is already contained in the number of distinct alleles $\mathcal{K}_n$, such that explicit knowledge of the configuration is not needed. In statistical terms, $\mathcal{K}_n$ is a sufficient statistic for θ. Indeed, the conditional probabilities $\mathrm{Prob}(\mathbf{C}(n) = \mathbf{c}(n) | \mathcal{K}_n = \kappa)$ are independent of θ. It holds that

$$\mathrm{Prob}(\mathbf{C}(n) = \mathbf{c}(n) | \mathcal{K}_n = \kappa) = \frac{n!}{\mathrm{coeff}(\theta(n), \kappa) \prod_j j^{c_j(n)} \prod_j c_j(n)!} . \tag{10.24}$$

Thus, when writing the probability $\mathrm{Prob}(\mathbf{C}(n) = \mathbf{c}(n) \wedge \mathcal{K}_n = \kappa)$ as a product of $\mathrm{Prob}(\mathbf{C}(n) = \mathbf{c}(n) | \mathcal{K}_n = \kappa)$ and $\mathrm{Prob}(\mathcal{K}_n = \kappa)$, only the last factor depends on θ. This implies that a maximum-likelihood estimate of θ can be based on the probability distribution of $\mathcal{K}_n$. Given an observation $\mathcal{K}_n = \kappa$, it can be shown that this maximum likelihood estimate is the solution of

$$\kappa = \sum_{i=0}^{n-1} \frac{\hat{\theta}}{\hat{\theta} + i} . \tag{10.25}$$

Furthermore, using the fact that under neutral evolution $\mathrm{Prob}(\mathbf{C}(n) = \mathbf{c}(n) | \mathcal{K}_n = \kappa)$ is independent of θ, one can construct a statistical test for the null hypothesis of neutral evolution. Unlikely configurations of $\mathbf{C}(n)$ given $\mathcal{K}_n$ would lead to a rejection of the null hypothesis. This idea is formalized in the Ewens-Watterson test.

Apart from the computation of the probability of an observed allele configuration under the infinite alleles model, Equation (10.23) allows one to deduce the expected number of singletons and the expected number of different alleles. The expected number of singletons in a sample of n genes is

$$E(C_1(n)) = \sum_{\text{all configurations } \mathbf{c}(n)} C_1(n) P\left(\mathbf{C}(n) = \mathbf{c}(n)\right) = \frac{n\theta}{n - 1 + \theta} . \tag{10.26}$$

The expected number of different alleles in a sample of n is

$$E(\mathcal{K}_n) = \sum_{\kappa=1}^{n} \kappa P(\mathcal{K}_n = \kappa) = \sum_{i=0}^{n-1} \frac{\theta}{\theta + i} , \tag{10.27}$$

where the last equality requires some arithmetical manipulations. In contrast, the formula given by Kimura and Crow (Eq. (10.20)) is a lower bound estimate for the number of simultaneously existing alleles in the population. In fact, $E(\mathcal{K}_n)$ is larger than $\underline{\kappa}$ only if n is sufficiently large. Furthermore, for large values of θ the expected number of alleles in a sample, $E(\mathcal{K}_n)$, may be smaller than the lower bound estimate for the number of alleles in the population, $\underline{\kappa}$ (see Table 10.2).

Finally, we note the variance of $\mathcal{K}_n$. It is

$$V(\mathcal{K}_n) = \sum_{i=0}^{n-1} \frac{\theta i}{(\theta + i)^2} .$$

10.7.2 Application

For the data presented in Table 10.1, some values of $E(C_1(n))$ and $E(\mathcal{K}_n)$ are shown in Table 10.2. The observed number of alleles is $\kappa = 9$. There are eight singletons and one set of three identical alleles. There are two possible configurations with $\kappa = 9$. One of them—the one observed—is $c_1(11) = (8, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0)$, the other possible one is $c_2(11) = (7, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0)$. From Equation (10.24) the conditional probabilities are

$$\text{Prob}(\mathbf{C}(11) = c_1(11)|\mathcal{K}_{11} = 9) = 0.25\,,$$
$$\text{Prob}(\mathbf{C}(11) = c_2(11)|\mathcal{K}_{11} = 9) = 0.75\,.$$

The maximum likelihood estimator for θ derived from Equation (10.25) is

$$\hat{\theta} = 20.73\,.$$

The observed allele configuration appears to agree well with the expected results for $E(\mathcal{K}_{11})$ and $E(C_1(11))$ for the parameter $\theta = 20$ (Fig. 10.12). On the other hand, 43 out of a total of 2,721 sites were polymorphic (or "segregating"), which is somewhat less than the 58.6 segregating sites expected under neutrality (Table 10.1). Is $\theta = 20$ a reasonable estimate for *D. melanogaster* and a stretch of 2.7 kb of its genome or do the data have to be explained by other evolutionary mechanisms than mutation and drift alone? After all, the two estimates of θ, obtained from Equation (10.21), which was $\hat{\theta} = 17.33$, and from Equation (10.27), which was $\hat{\theta} = 20.73$, are similar but not identical.

Table 10.2. Numerical values for the expected number of alleles, number of segregating sites and singletons of the data presented in Table 10.1.

θ	$\underline{\kappa}$†	$E(\mathcal{K}_{11})$	$V(\mathcal{K}_{11})$	S‡	$E(C_1(11))$
0.02	1.020	1.058	0.057	0.586	0.022
2.00	3.000	4.206	1.947	5.858	1.833
6.12	7.120	6.629	2.234	17.925	4.176
20.0	21.00	8.945	1.550	58.579	7.333

†lower bound estimate according to Equation (10.20).
‡expected number of segregating sites according to Equation (11.8).

We will come back to the problem of testing the significance of the difference between the neutral expectation and observation in Chapters 11 and 12.

10.8 Selection

The neutral theory of molecular evolution should not detract us for too long from the really interesting question, namely how is adaptation of a species to certain environmental conditions possible? Most answers contain in one way or the other the concept of an “advantageous mutation” [76]. An organism carrying an advantageous mutation, i.e. a genetic constitution which makes it better adapted to the given environmental conditions, will on average leave more offspring to the next generation than its competitors. Such an organism is said to have an increased fitness. In a very large population, which is the case envisaged by Fisher, the dynamics of such an advantageous mutation can essentially be treated in a deterministic framework. The following model captures the salient features of the spread of an advantageous mutation within a population.

Let x and y be the relative frequencies of the two alleles “advantageous mutation” (allele B) and “wild-type” (allele b), respectively. As always in a two-allele model, $x + y = 1$. For diploid organisms, i.e. organisms which have two copies of each chromosome, let the fitnesses of the three genotypes bb, Bb and BB be

BB	Bb	bb
$1+2s$	$1+s$	1

In other words, organisms with two copies of the advantageous mutation have a fitness difference of $2s$ compared to those which have two wild-type chromosomes. The dynamics of the B-type chromosomes is described by a difference equation which accounts for the change in frequency of type B when passing from one generation to the next

$$\Delta x_t = x_{t+1} - x_t = \frac{(1+2s)x_t^2 + (1+s)x_t y_t}{(1+2s)x_t^2 + 2(1+s)x_t y_t + y_t^2} - x_t\,. \tag{10.28}$$

Note, that the denominator on the righthand side represents the average fitness of the population, where the average is taken over the three possible genotypes. In fact, this term is called *mean fitness*. The difference equation (10.28) is approximated by an ordinary differential equation, the so-called deterministic selection equation

$$\frac{dx}{dt} = sx(t)(1 - x(t)) \tag{10.29}$$

with the initial condition

$$x(0) = \epsilon\,.$$

The solution of this differential equation is

$$x(t) = \frac{\epsilon}{\epsilon + (1-\epsilon)\exp(-st)}\,. \tag{10.30}$$

Since the advantageous mutation is initially present only on a single chromosome, one typically assumes $\epsilon = 1/(2N)$, where N is the (effective) population size. The time which the advantageous mutation requires to increase from frequency ϵ to $1-\epsilon$ is obtained by solving

$$1 - \epsilon = \frac{\epsilon}{\epsilon + (1-\epsilon)\exp(-st)}$$

for t. This is

$$t_{(\epsilon)}^{(1-\epsilon)} = \frac{2}{s} \log \frac{1-\epsilon}{\epsilon} \approx \frac{2}{s} \log(2N) . \qquad (10.31)$$

It can be shown that the righthand side of Equation (10.31) is a good approximation of the fixation time, t_{fix}, for an advantageous allele also in a non-deterministic framework.

For the above fitness regime, mean fitness simplifies to

$$\bar{w}(t) = 1 + 2sx(t) .$$

Since $x(t)$ is a monotonically increasing function in t, mean fitness also has this property. In fact, Fisher's fundamental theorem of natural selection states that "mean fitness of a population increases". In other words, evolution is a hill-climbing process towards a fitness peak. However, it has to be emphasized that this is true only for the deterministic dynamics of infinitely large populations. In finite sized populations, positive selection and genetic drift act antagonistically. While a positively selected, newly arisen, allele is still rare in a population, there is a non-zero probability that this allele will be lost again by drift. It can be shown [65] that the fixation probability of an advantageous allele is goverend by its selective advantage and that it is approximately

$$\mathrm{Prob}_{\text{fix}} = 2s . \qquad (10.32)$$

A positively selected allele initially increases at a rate s per generation (see Eq. (10.30)), while diversity is lost due to drift at a rate of $1/(2N)$ per generation (see Eq. (10.12)). Crucial for positive selection to prevail as an evolutionary force over drift is the relation

$$2Ns > 1 . \qquad (10.33)$$

Thus, rather than the net fitness advantage alone, it is its relation to population size that is decisive for the possibility of adaptation. In sufficiently large populations, there is a chance even for weakly selected alleles to become fixed. The issue whether the "normal" mode of genome evolution is evolution under weak positive selection is the matter of an old debate [153, 115]. One of the problems, implicated by Equation (10.33), is that the action of weak selection is hard to prove in laboratory experiments. The quest remains to distinguish the action of selection from "molecular noise". This difficulty has been called the "uncertainty principle of molecular evolution" [235, 242].

The combined action of mutation, selection, and drift introduces rich dynamics into the evolutionary system which allows for a variety of non-trivial equilibria depending on the relative magnitudes of the selection coefficient, dominance effects, mutation rate, and population size [32, 14]. A good part of the mathematical foundation for these models has been laid out by Haldane, Fisher, and Wright in the first half of the past century [100, 101, 76, 262].

10.9 Summary

This chapter is concerned with the dynamics of genes in populations. The neutral Wright-Fisher model provides a convenient framework for describing such dynamics. This model is based on a population of constant size that evolves from one generation to the next by sampling the genes with replacement. Neutrality then simply corresponds to the fact that all genes are equally likely to get transferred to the next generation. Changes in allele frequencies are random and said to be due entirely to genetic drift. Given such a model without mutation, any existing genetic diversity will eventually be lost from the population. In other words, there will be only one allele for each gene. The probability that a certain allele replaces all others, i.e. is fixed, is equal to the initial frequency of the allele. If we add mutation to the model, the mutation process can be described in three different ways: (i) under the finite alleles model each mutation leads to a random switch between k alleles; (ii) under the infinite alleles model each mutation generates a new allele; (iii) under the infinite sites model each mutation affects a different site along a stretch of sequence. Mutations create new alleles, while drift removes alleles from the gene pool and these two factors can come into a mutation drift balance. If mutation is balanced by drift, it is possible to compute the number of alleles expected to occur in a population as well as the probability of drawing a pair of distinct alleles, also known as the genetic diversity. Since the number of alleles can easily be observed, a comparison between observed and expected allele configurations is used as a test of neutral evolution.

10.10 Further Reading

This chapter is mainly based on the comprehensive textbook by Hartl and Clark [106] as well as on the concise primer by Gillespie [87]. Either one of these books should be consulted for further details and references on the foundations of population genetics. Ewens [65] gives a mathematically advanced treatment of Markov chain theory and diffusion models in population genetics. Finally, Provine describes the history of population genetics by way of an illuminating biography of Sewall Wright [204].

10.11 Exercises and Software Demonstration

10.1. The `bioinformer` software contains under `Evolution` → `Drift` a program that shows forward in time simulations. Work through the tutorial for that program in Section A.4.2.

10.2. In the earliest attempt to test the validity of the neutral hypothesis using bacterial populations, a *virtual* heterozygosity of $H = 0.242$ was measured in *E. coli* [182]. Why was the genetic diversity in *E. coli* referred to as "virtual" heterozygosity?

10.3. In the *E. coli* study a mutation rate of $\mu = 10^{-8}$ was assumed. Which population size follows from this under the neutral infinite alleles model?

10.4. Based on considerations independent of the genetic diversity, the population size of *E. coli* was estimated to be 10^{10}. Given this estimate, what is the genetic diversity, H, expected under the neutral infinite alleles model?

10.5. The population size relevant for population genetics is usually referred to as *effective* population size, N_e. It may differ widely from a population's head count. Essentially, N_e is the size an idealized population would have, given the effect of drift on its allele distribution [106, p. 289]. A population study of *E. coli* has come to the conclusion that $N_e = 1.8 \cdot 10^8$, which is surprisingly small [107]. What is the value of H expected from this estimate of population size?

10.6. Show that allele frequencies do not change between generations when $N \to \infty$, i.e. show that the Hardy-Weinberg law holds if the population size is infinitely large. Hint: Start from Equation (10.1).

10.7. Consider a haploid population with $N = 50$. What is the probability that two randomly selected alleles had their last common ancestor exactly 1,000 generations ago?

10.8. In Section 10.7.1 a value of $\hat{\theta} = 20.73$ was found based on the assumption that there are nine alleles in the sample of eleven *Adh* sequences of *D. melanogaster*. Remember that there are ten alleles, if indel ∇_3 is included in the analysis. Estimate θ on the basis of ten alleles.

10.9. Consider a diploid population with k alleles. Show that the proportion of homozygotes is minimal if all alleles are equally frequent.

11

Genes in Populations: Backward in Time

> *The circle of ideas that has come to be known as the coalescent has proved to be a useful tool in a range of genetical problems, both in modeling biological phenomena and in making statistical sense of the rich data now available.*
>
> *John F. C. Kingman [149, p. 1461]*

Intuitively, we imagine evolution as moving forward in time. Starting perhaps four billion years ago in a marine RNA world, life evolved *via* the ancestor of all cells, the invention of multicellularity, the conquest of land and air to the present, constantly changing, overwhelming diversity. If we restrict our attention to the microevolution of single populations, we have seen in Chapter 10 that the Wright-Fisher model, which in its original formulation moves forward in time, is useful for capturing the dynamics of genes. In this chapter we turn the Wright-Fisher model on its head to look backward in time. This inversion of the arrow of time forms the basis of almost all contemporary investigations of the dynamics of genes in populations. Such investigations encompass a wide range of problems, in particular understanding the distribution of polymorphisms along and between human genomes [244].

11.1 Individuals' Genealogies *vs.* Gene Genealogies

We all have $2^1 = 2$ parents, $2^2 = 4$ grandparents, $2^3 = 8$ great-grandparents, and so on. In other words, the number of our ancestors increases exponentially as we move backward in time. This means that your ancestors rapidly also become the ones of any other person you meet. If we assume panmixis and constant, sufficiently large population size, the first common ancestor of all members of a sexually reproducing population is expected to appear after $\log_2 N$ generations, where N is the population size [208]. For humans this would place the universal ancestor in the Middle Ages. Perhaps even more surprising than this extremely rapid appearance of universal ancestors is the discovery that after $1.77 \log_2 N$ generations the ancestral population has only two kinds of members: approximately 80% are universal ancestors, and the remaining 20% have left no descendants in the present [208].

Figure 11.1A shows the simulated genealogy of a single member of a sexual diploid population over 20 generations. Going back just five generations we reach the first two individuals that are the ancestors of the entire population. The fact that two such most recent universal ancestors turn up in the same generation is due to chance, it might have been just a single individual. Going back further in time we enter a zone, where individuals might be ancestors of all, none, or part of the present population. This only lasts for three generations, however, in g_{12} all individuals are either ancestors of the entire present day population, or have left no descendants at all. This situation now remains stable *ad infinitum*. The only fluctuations we observe from now on are in the distribution of the numbers of universal ancestors and "universal losers".

Now, real populations are not panmictic and do not have constant size. Nevertheless, a realistic simulation of human history led to the conclusion that universal ancestors of all of humanity lived around 1,400 B.C. in Asia. Further, from the year 5,353 onward into the past we all have identical ancestors [208].

At this point we come to the important distinction between the genealogy of individuals just discussed and that of genes. In contrast to individuals, each gene has only a single ancestor. This places common gene ancestry much further back in time than common individual ancestry. As a consequence, it is not certain that a single gene of a random individual living today can be traced back to our Asian ancestor of not so long ago. Figure 11.1B traces the ancestry of the two homologous genes contained in a single individual. We have to wait for 18 generations until they find their first common ancestor.

In order to learn more about the dynamics of genes in time we now return to the Wright-Fisher model of gene evolution already encountered in Section 10.4.

11.2 Forward *vs.* Backward in Time

The Wright-Fisher model (Section 10.4) is forward in time in the sense that a quantity in generation t is a function of the same quantity in generation $t-1$. Let us look again in detail at the structure of the Wright-Fisher model. The first thing to realize is that for many evolutionary scenarios there is no need to explicitly model diploidy or sexual dimorphism as was done in Figures 11.1A and 11.1B. Figure 10.5 illustrated how through resampling with replacement the correspondingly uniform gene pool in generation $t - 1$ is transmitted to the gene pool in generation t. The example population depicted consists of $N = 6$ diploid organisms and hence we simply model $2N = 12$ homologous genes.

Figure 11.2A shows all lines of descent for our example population throughout a simulation run of 15 generations. Notice that due to sampling with replacement five genes get lost in the transition from generation 1 to generation 2, while three genes each leave one descendant, three genes leave two descendants and one leaves three descendants. Otherwise, Figure 11.2A is too tangled for easy reading. Figure 11.2B shows its untangled version, which allows tracing of individual lineages. As we pick any gene, e.g. from generation g_{10}, and follow its fate through the generations forward in time, it is never clear whether it will go extinct or show up in the present population. Conversely, if we concentrate on the genes in the present generation, g_{15}, and follow their fate backward in time, we will, given enough time, eventually hit the first gene that was the ancestor of the entire population. In Figure 11.2 the

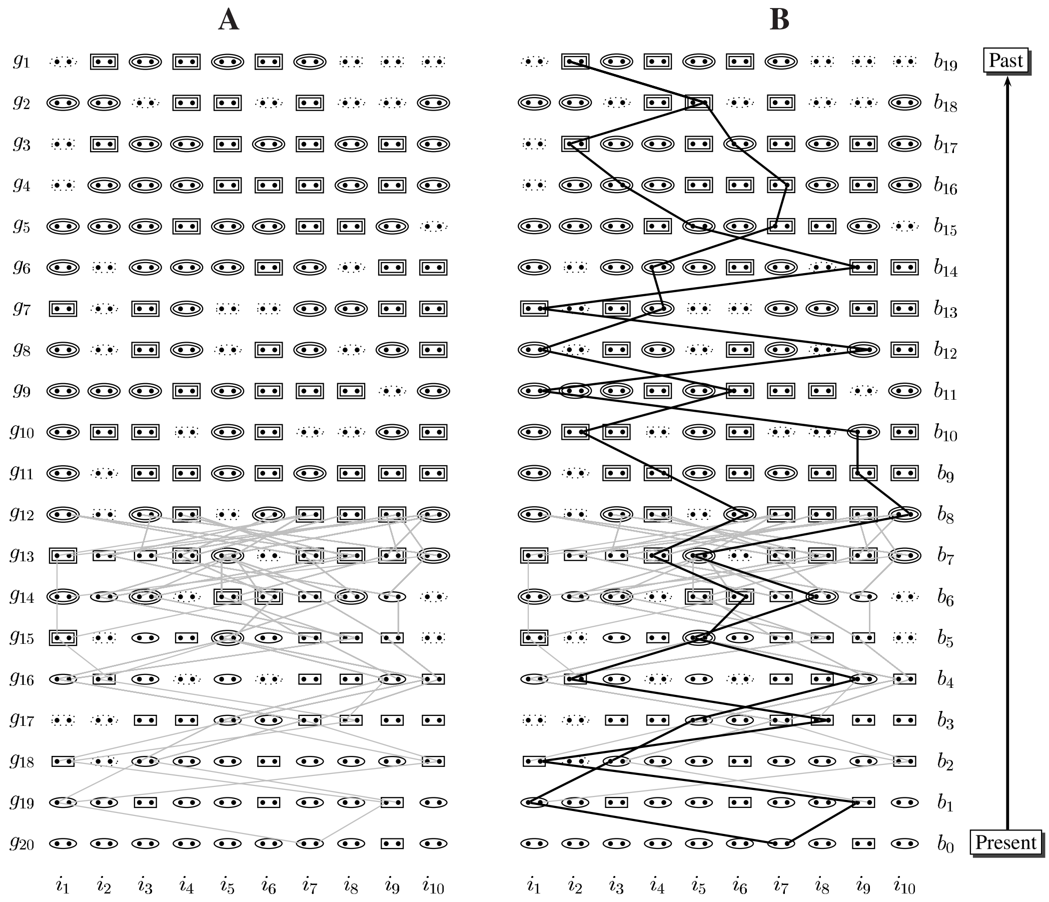

A
B
g_1 g_2 g_3 g_4 g_5 g_6 g_7 g_8 g_9 g_{10} g_{11} g_{12} g_{13} g_{14} g_{15} g_{16} g_{17} g_{18} g_{19} g_{20}
i_1 i_2 i_3 i_4 i_5 i_6 i_7 i_8 i_9 i_{10}
b_{19} b_{18} b_{17} b_{16} b_{15} b_{14} b_{13} b_{12} b_{11} b_{10} b_9 b_8 b_7 b_6 b_5 b_4 b_3 b_2 b_1 b_0
Past
Present

Fig. 11.1. **A**: The number of ancestors of any sexual organism increases exponentially along its genealogy. ●: Gene; box: male; ellipse: female; double line: universal ancestor; dotted line: no descendants in present generation; g_i: i-th generation; b_i: going back i generations. To limit clutter, lineages are only drawn for eight generations. **B**: The same as **A** but with superimposed gene ancestry shown in black lines.

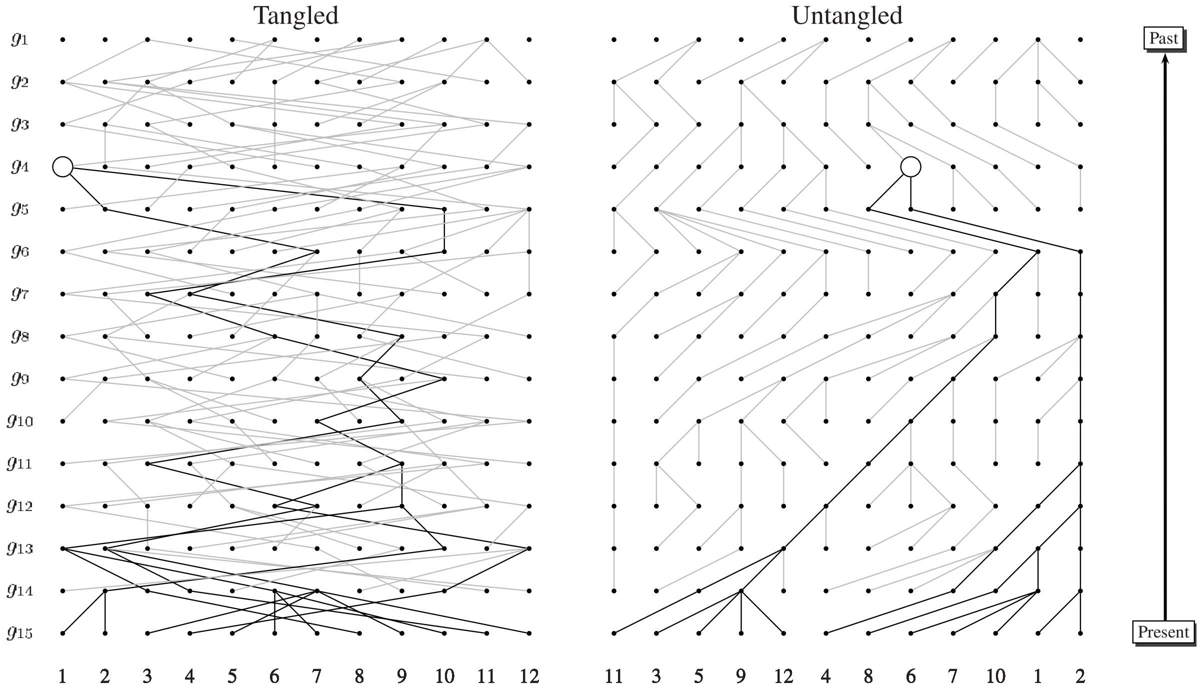

Tangled
Untangled
g_1
g_2
g_3
g_4
g_5
g_6
g_7
g_8
g_9
g_{10}
g_{11}
g_{12}
g_{13}
g_{14}
g_{15}
1 2 3 4 5 6 7 8 9 10 11 12
11 3 5 9 12 4 8 6 7 10 1 2
Past
Present

Fig. 11.2. Lines of descent of 12 genes for 15 generations under the Wright-Fisher model of evolution, where generation t is produced from generation $t - 1$ by sampling with replacement. ○ indicates the most recent common ancestor; black lines are the lineages of extant genes; gray lines show extinct lineages.

most recent common ancestor of the population is marked by ○. Clearly, this most recent common ancestor also has ancestors; in fact, as we go back in time, we find in each generation beyond the most recent common ancestor exactly one gene from which the members of our entire example population have descended. Among these, the most recent common ancestor is very special. The reason for this is that genetic events, such as mutations, that might differentiate the members of our population must have occurred since their descent from this last common ancestor. Conversely, any event along a lineage leading from somewhere in the past to the most recent common ancestor has equally affected all members of the population and is therefore invisible.

However, we usually analyze samples of genes taken from large populations. The genealogy of any random sample is a subgenealogy of the population genealogy. An example for the sample of genes 1, 2, 4, and 8 is shown in Figure 11.3.

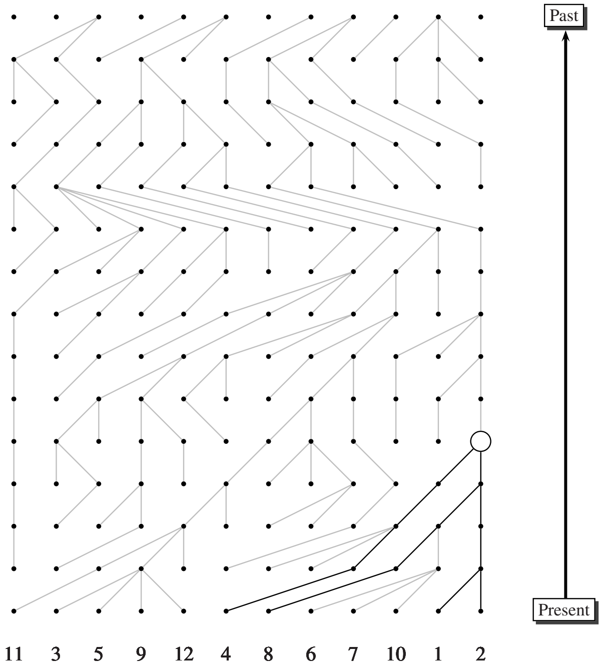


Fig. 11.3. Lines of descent for a sample of $n = 4$ genes form a subgraph of the population genealogy shown in Figure 11.2. ○ indicates the most recent common ancestor of the sample. It coincides with the most recent common ancestor of the population with probability $\frac{n-1}{n+1} = 0.6$ [192].

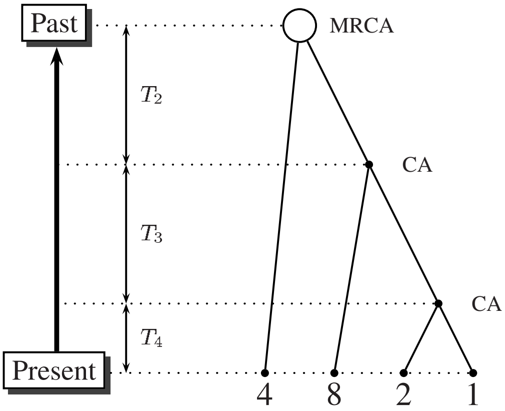


Fig. 11.4. Coalescent of four genes sampled from the population shown in Figures 11.2 and 11.3. CA: coalescence event; ○: most recent common ancestor (MRCA); T_i: time interval in which the coalescent consists of i lineages.

11.3 The Coalescent

Figure 11.4 displays the genealogy of the sample of genes 1, 2, 4, and 8 extracted from their population context (Fig. 11.2). It is a binary tree rooted on the most recent common ancestor. When moving from the tips of the tree, representing the sampled genes, to its root, the most recent common ancestor, we pass internal nodes where two lineages coalesce into their most recent common ancestor. Such a fusion of two lineages is also known as a coalescence event. The complete topology of coalescence events is called the *coalescent* and the theory built around this data structure is referred to as *coalescent theory*. It was first proposed in the early 1980s [147, 148, 119, 239, 149] and has in the meantime developed into a subdiscipline in theoretical population genetics in its own right [120, 192].

Since a coalescent tree is a rooted bifurcating tree, it consists of $2n - 1$ nodes, where n is the size of the sample. In other words, construction of such a tree takes time proportional to the sample size. This leads to a great efficiency gain compared to forward simulation, where entire populations need to be represented in computer memory and a possibly very large number of lineages is traced. The vast majority of these lineages do not contribute genes to the sample from which the statistic of interest is calculated (Fig. 11.2). Moreover, the constellation of genes in generation g_t clearly depends on their constellation in generation g_{t-1}. This leads to autocorrelated measurements, such as those shown in Figure 10.6. However, statistical analyses are usually based on the assumption of independence, which is best implemented by constructing a separate coalescent for each measurement. Apart from generating independent samples, coalescent simulations have the huge advantage of just tracing the history of a gene sample rather than that of the entire population.

A collection of coalescent trees is shown in Figure 11.5. Notice that the time to the most recent common ancestor, i.e. the height of these genealogies, appears to be only weakly correlated with sample size. We will quantify this relationship in

Section 11.5; but before we do so, we shortly comment on the differences between coalescent and phylogenetic trees.

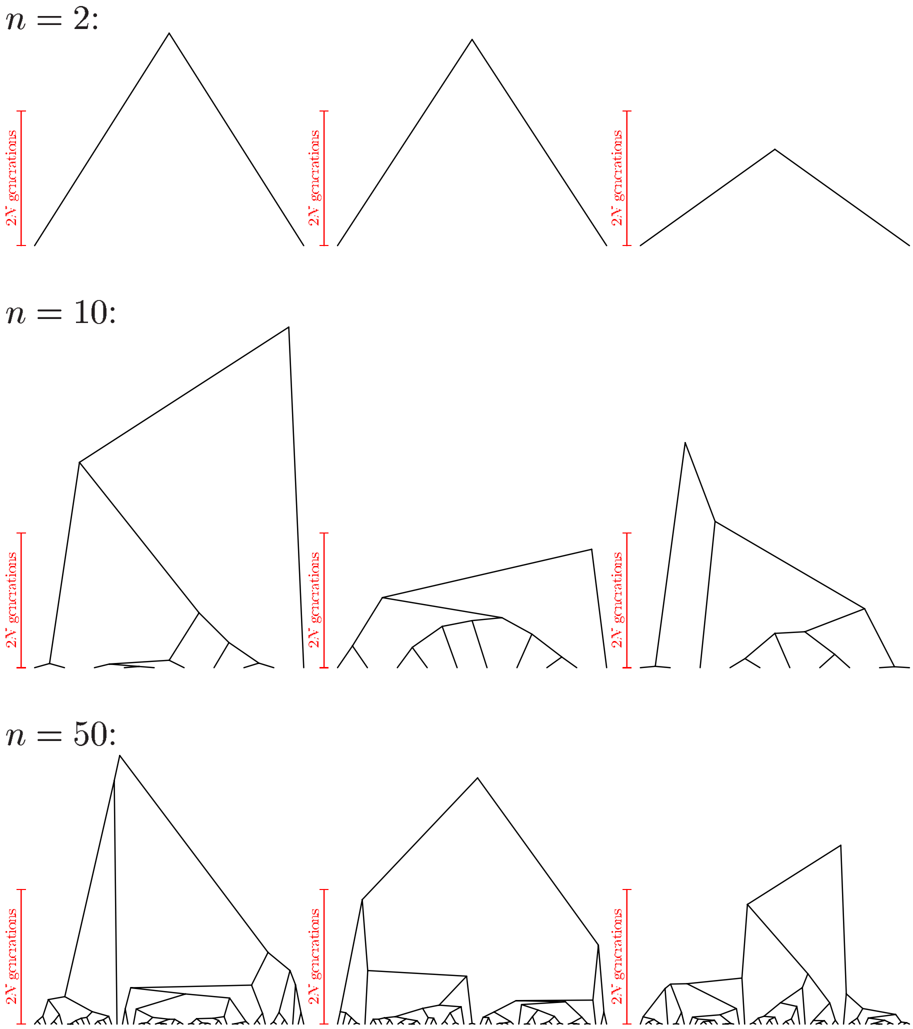

$n = 2$:
2N generations
2N generations
2N generations
$n = 10$:
2N generations
2N generations
2N generations
$n = 50$:
2N generations
2N generations
2N generations

Fig. 11.5. Examples of neutral gene genealogies in a haploid population for sample sizes 2, 10, and 50. Time is measured in $2N$ generations, where N is the (usually unknown) size of the population.

11.4 Coalescent *vs.* Phylogenetic Trees

Coalescent and phylogenetic trees are both bifurcating trees, but their construction and application is very different. Phylogenies are usually reconstructions of the one "true" evolutionary history of a set of species. In contrast, coalescent trees are random histories of genes sampled from a single population. Their construction is based on model parameters, which may or may not have been obtained from empirical data. Table 11.1 summarizes the main differences between these two important evolutionary trees.

Table 11.1. Comparison between the coalescent and the phylogeny of a sample of genes.

Feature	Phylogeny	Coalescent
Level of comparison	Inter-species gene history	Intra-species gene history
Purpose	Reconstruct the true species history	Simulate sets of potential gene histories
Observation of interest	Tree topology	Frequency spectrum and distribution of segregating sites
Data source	Comparative data	Model parameters
Multiplicity	Single	Many

11.5 The Infinite Sites Model and the Number of SNPs

Our first application of coalescent theory concerns the number of single nucleotide polymorphisms (SNPs) expected under neutrality in a sample of n genes. Among geneticists SNPs are also known as segregating sites. The number of segregating sites, which can easily be observed from alignments, is a simple function of the unobservable quantity $N\mu$, the product of population size, N, and mutation rate, μ. Our derivation of the number of segregating sites is based on the infinite sites model, according to which no two mutations hit the same nucleotide (Fig. 11.6). This is a reasonable assumption, given that the majority of observed SNPs are bi-allelic (cf. Table 10.1). Under the infinite sites model the number of SNPs in a sample is equal to the number of mutation events in the corresponding coalescent. Hence, the expected number of SNPs is equal to the rate of mutation times the total length of the coalescent. We are now going to derive basic properties of the total length of a coalescent, i.e. the sum of all branch lengths.

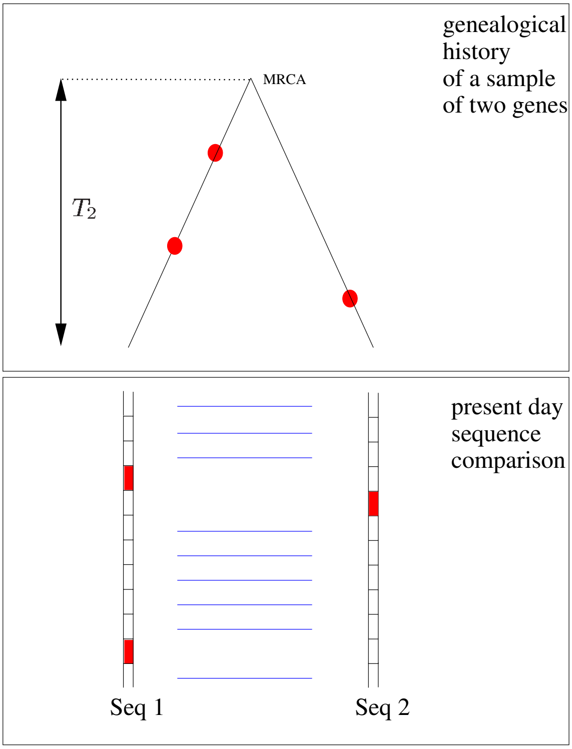

genealogical history of a sample of two genes
MRCA
T_2
present day sequence comparison
Seq 1
Seq 2

Fig. 11.6. *Top*: Genealogy for sample size $n = 2$. Mutations are depicted as dots along the branches of the genealogy. *Bottom*: Comparison of the two sequences connected by the genealogy shown on top. Matching positions are shown as vertical lines. Under the infinite sites model the number of (unobservable) mutations is equal to the number of observable segregating sites (SNPs) in the sample. For a given coalescence time, $T_2 = t_2$ say, the number of segregating sites is a Poisson-distributed random variable with parameter $\lambda = 2\mu t_2$, where μ is the mutation rate per site per generation. MRCA: most recent common ancestor.

11.6 Mathematical Properties of the Neutral Coalescent

11.6.1 Tree Depth, Tree Size and the Number of Segregating Sites

Consider again the coalescent shown in Figure 11.4, which has time intervals marked. Each coalescence event reduces the number of lineages by one. We call the time interval in which the coalescent consists of i lineages T_i. Our first aim is to compute the average length of T_i. For this we need the probability at any point in time that no coalescence took place in the previous generation. Consider a gene pool of size $2N$; that is, from each of N members of a diploid population we include its two copies of a particular gene, α-globin say, in this pool. We move backward in time from generation t to generation $t - 1$. As we do so, we invert the traditional time-forward metaphor of genes in generation $t - 1$ leaving descendants in generation t

and imagine instead that genes in generation t pick their ancestors in generation $t-1$. Two genes pick a particular common ancestor with probability $(1/(2N))^2$. There are $2N$ possible common ancestors, hence the probability of two lineages coalescing is $1/(2N)$ in each generation and the probability of them *not* coalescing is $1-1/(2N)$.

We now turn our attention to the sample of n genes. Obviously, the first gene has some ancestor in the previous generation. The probability that the second gene has a different ancestor is simply the probability of not coalescing,

$$1-\frac{1}{2N}.$$

The probability that three genes each have different ancestors in the preceding generation is

$$(1-\frac{1}{2N})\cdot(1-\frac{2}{2N}).$$

In general, the probability that n genes have no common ancestor is

$$\left(1-\frac{1}{2N}\right)\left(1-\frac{2}{2N}\right)\ldots\left(1-\frac{n-1}{2N}\right)\approx 1-\frac{1}{2N}-\frac{2}{2N}-\ldots-\frac{n-1}{2N}.$$

The approximation is obtained by assuming that population size is large and that therefore terms of order N^{-2} or less may be neglected. The probability, p_n, of a coalescence occurring is the complement of it not occurring, i.e.

$$p_n=\frac{1+2+\ldots+(n-1)}{2N}=\frac{n(n-1)}{4N}.$$

Thus, going backward in time, for each generation there is a probability p_n for a coalescence to occur. The probability that the first coalescence occurs after exactly $t+1$ generations is therefore $(1-p_n)^t\cdot p_n$. This means that coalescence times are geometrically distributed with parameter p_n. Hence, the expected time it takes for n lineages to coalesce to $n-1$ lineages is $(1-p_n)/p_n$. Implicit is the assumption that population size, N, remains constant over time. Furthermore, if N is large compared to the sample size n, then $(1-p_n)/p_n\approx 1/p$ and the geometric distribution is well approximated by an exponential. By generalizing this to the waiting time for a coalescence event in a sample of size i, we can write

$$E\{T_i\}=\frac{4N(1-\frac{i(i-1)}{4N})}{i(i-1)}\approx\frac{4N}{i(i-1)}.$$

Each interval T_i is then approximately exponentially distributed with parameter $p_i=i(i-1)/(4N)$. The *tree depth* is the expected time for all lineages to coalesce, i.e. the expected time to the most recent common ancestor (MRCA; T_2 in Figure 11.6). It can be computed by applying the rule that the expectation of a sum of random variables is equal to the sum of the expectations:

$$E\{T_{\mathrm{MRCA}}\}=\sum_{i=2}^{n}E\{T_i\}=\sum_{i=2}^{n}\frac{4N}{i(i-1)}=4N\left(1-\frac{1}{n}\right).$$

In other words, the expected time to the most recent common ancestor varies by a factor of just 2 between sample size $n = 2$ and $n = \infty$. This is the reason why the heights of the coalescent trees displayed in Figure 11.5 do not increase much as we increase the sample size by more than an order of magnitude.

Note, however, that the above property applies only to the expected time of the MRCA, $E\{T_{\mathrm{MRCA}}\}$. Nothing has been said so far about the random variable T_{MRCA} itself. In fact, it is a sum of exponentially distributed random variables T_i each with a different parameter $p_i < p_j$, if $i < j$. In order to compute the variance of T_{MRCA}, we note that the random variables $(T_i)_i$ are independent. The length of time interval T_i depends only on the parameters i and N, but not on time intervals T_j, $j \neq i$. Therefore, the variance is

$$V\{T_{\mathrm{MRCA}}\} = \sum_{i=2}^{n} V\{T_i\} = \sum_{i=2}^{n} \left(\frac{4N}{i(i-1)} \right)^2 . \tag{11.1}$$

For large sample sizes n we take the limit $n \to \infty$ of the summation in Equation (11.1) and obtain the approximation

$$V_{n=\infty}\{T_{\mathrm{MRCA}}\} = (4N)^2 \left(\frac{\pi^2}{3} - 3 \right) \approx (4N)^2 \cdot 0.29 .$$

An approximate expression for finite n can be found in the following way. We replace the sum on the righthand side in Equation (11.1) by a definite integral ranging from $3/2$ to $(n + 1/2)$. We then add a constant, such that for large n the limiting values of this integral and the above approximation become identical, and obtain

$$V_n\{T_{\mathrm{MRCA}}\} \approx (4N)^2 \left(\frac{\pi^2}{3} - 3 - \frac{8n}{4n^2 - 1} - 2 \log \frac{2n-1}{2n+1} \right) . \tag{11.2}$$

We now turn to the computation of the complete branch length, T_c, of the coalescent tree. This is the sum of the lengths of all branches contained in the tree. It is

$$T_c = \sum_{n=2}^{n} i\, T_i .$$

Again, the expectation of T_c, $E\{T_c\}$, is obtained from the sum of the individual expectations, $E\{T_i\}$:

$$E\{T_c\} = \sum_{i=2}^{n} i\, E\{T_i\} = 4N \sum_{i=2}^{n} \frac{1}{i-1} = 4N \sum_{i=1}^{n-1} \frac{1}{i} . \tag{11.3}$$

Using Euler's constant $\gamma \approx 0.577$, this equation can be approximated by

$$E\{T_c\} \approx 4N \left(\gamma + \frac{1-\gamma}{n-1} + \log(n-1) \right) . \tag{11.4}$$

To compute the variance of T_c, we note that $(T_i)_i$ are independent random variables, and so are $(i \cdot T_i)_i$. Therefore, the variance of T_c is obtained from the individual variances as well:

$$V\{T_c\} = \sum_{i=2}^{n} i^2 V\{T_i\} = (4N)^2 \sum_{i=2}^{n} \frac{1}{(i-1)^2} \,. \qquad (11.5)$$

For large n we take the limit $n \to \infty$ of the summation in Equation (11.5) and find

$$V_{n=\infty}\{T_c\} = (4N)^2 \frac{\pi^2}{6} \approx (4N)^2 \cdot 1.64 \,.$$

The variance for finite n is approximated by

$$V_n\{T_c\} \approx (4N)^2 \left(\frac{\pi^2}{6} - \frac{1}{n - 1/2} \right) \,. \qquad (11.6)$$

Figure 11.7 shows a plot of the expectation and variance of T_{MRCA} and T_c *vs.* sample size n.

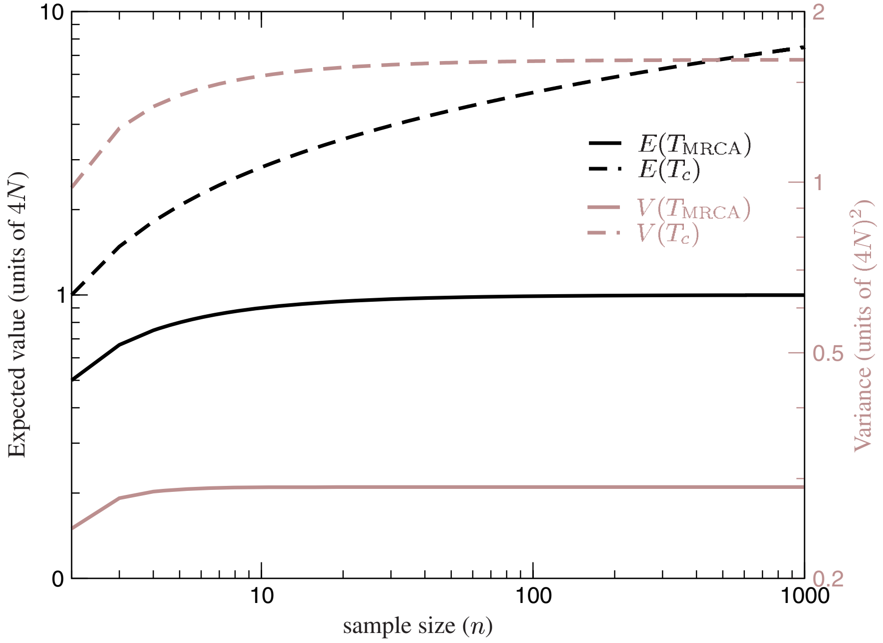


Fig. 11.7. Expectation and variance of T_{MRCA} (solid) and T_c (dashed). Expectation is plotted in units of $4N$ (left y-axis, black), and variance in units of $(4N)^2$ (right y-axis, gray).

Having discussed coalescent events, we now consider mutation events on the tree. We noted above that mutations accumulate along the branches according to a Poisson distributed random variable with parameter μ per unit time. Under the infinite sites model this number is identical to the expected number of polymorphisms (or SNPs) in an alignment. For fixed time $T_c = t_c$, the conditional expectation of the number of mutations on the tree, and therefore the number of segregating sites, is

$$E\{S_n|T_c = t_c\} = \mu\, t_c\,.$$

The conditional expectation $E\{S_n|T_c\}$, which is itself a random variable, has expectation $E\left\{E\{S_n|T_c\}\right\} = E\{S_n\}$. Thus,

$$E\{S_n\} = \int_t E\{S_n|T_c = t\} dP_{T_c}(t) = \int_t \mu t\, dP_{T_c}(t) = \mu \int_t t\, dP_{T_c}(t) = \mu\, E\{T_c\}\,, \tag{11.7}$$

where the integral is evaluated with respect to the (unknown) probability density P_{T_c} of T_c. Inserting the righthand side of Equation (11.3), one obtains

$$E\{S_n\} = \theta \sum_{i=1}^{n-1} \frac{1}{i}\,, \tag{11.8}$$

where $\theta = 4N\mu$. Equation (11.8) is due to Watterson [258] and is one of the most widely applied results in population genetics.

To calculate the second moment, $E\{S_n^2\}$, we use the fact that [135, p.9]

$$E\{S_n^2\} = E\left\{E\{S_n^2|T_c\}\right\}\,.$$

Furthermore, using $E\{S_n^2|T_c = t\} = V\{S_n|T_c = t\} + E^2\{S_n|T_c = t\}$, the righthand side in the above equation can be written as

$$E\left\{E\{S_n^2|T_c\}\right\} = \int_t E\{S_n^2|T_c = t\}\, dP_{T_c}(t) = \int_t \mu t + (\mu t)^2\, dP_{T_c}(t)\,.$$

Therefore,

$$E\left\{E\{S_n^2|T_c\}\right\} = \mu E\{T_c\} + \mu^2 E\{T_c^2\}\,.$$

Finally, the variance of S_n is

$$V\{S_n\} = \mu E\{T_c\} + \mu^2 E\{T_c^2\} - \mu^2 E^2\{T_c\} = \mu E\{T_c\} + \mu^2 V\{T_c\} \tag{11.9}$$

or, in terms of θ,

$$V\{S_n\} = \theta \sum_{i=1}^{n-1} \frac{1}{i} + \theta^2 \sum_{i=1}^{n-1} \frac{1}{i^2}\,.$$

Note that the expected number of segregating sites and its variance are only functions of μ and T_c, and do not depend on the topology of the underlying tree. In fact, for given n and given time intervals $T_i = t_i$, $2 \le i \le n$, any two topologies have the same total branch length $t_c = \sum_i i\, t_i$ and therefore the same expected number of mutations. In contrast, tree topology can influence the *frequency spectrum* of mutations. Consider the two topologies shown in Figure 11.8, each with one mutation.

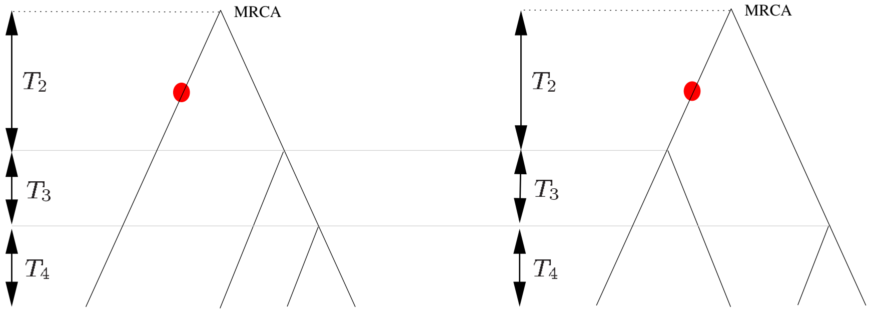


Fig. 11.8. Depending on the topology, the frequency spectrum of polymorphisms may change. The *left panel* shows one singleton mutation, because the mutation (dot) is located on a branch leading to a leaf of the tree. In contrast, in the *right panel* the mutation is located on an internal branch and, hence, there is no singleton mutation. The combined outer branch lengths are $4T_4 + 2T_3 + T_2$ and $4T_4 + 2T_3$, respectively. The level of heterozygosity is influenced as well. In the left topology the average number of pairwise differences is 0.5, in the right topology it is $0.6\bar{6}$.

The left panel shows a singleton mutation, as it affects only a single chromosome in the sample. In contrast, the mutation on the right genealogy affects two of the four sampled genes. Singleton mutations are the ones which occur only in a single chromosome in the sample. Singletons can only accumulate on outer branches of the tree. Therefore, the combined branch lengths of all outer branches determines the expected number of singleton mutations in the sample. The two trees in Figure 11.8 have different combined outer branch lengths, yielding different numbers of singleton mutations observed in the sample.

In contrast to the number of segregating sites (or SNPs), heterozygosity is a measure of genetic variability which depends on the frequency spectrum of mutations. As we will see in Chapter 12, this fact is exploited to construct statistical tests of the neutral theory.

In the step leading to Equation (11.8) we had added mutations to the model and it is instructive to consider the relationship between mutations along the coalescent and the resulting SNP patterns. Mutations affecting many chromosomes tend to have occurred early in the history of the sample as illustrated by the mutations shown as dark gray dots on the left panel of Figure 11.9. In contrast, singleton mutations can only have occurred on an outer branch of the genealogy (Fig. 11.9, left panel, black dots). Any mutation occurring before the most recent common ancestor of a sample leaves no trace in the SNP pattern (Fig. 11.9, left panel, light gray dots). While the action of drift may lead to random fluctuations in coalescent times T_i and tree topology, natural selection leads to systematic changes of these quantities (Fig. 11.9, middle and right panels).

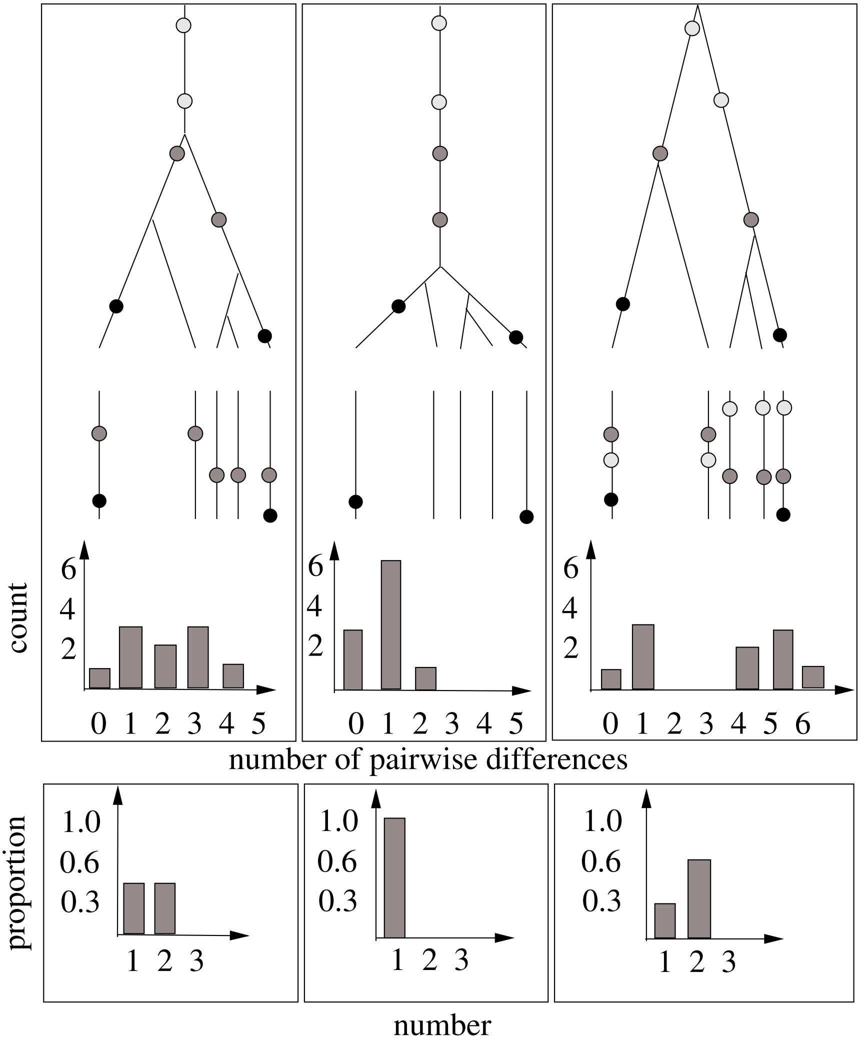


Fig. 11.9. *Top*: Gene genealogies and number of pairwise differences for different selection regimes. Displayed are mutations (*dots*) on the coalescent for sample size $n = 5$ and the corresponding SNP patterns. The chromosomes are drawn vertically. Below them are plots of the number of pairwise differences, also called the mismatch distribution, for each panel. Under directional selection (*middle panel*) average heterozygosity is lower than in the neutral reference case (*left panel*), because directional selection reduces the number of pairwise differences. In contrast, balancing selection (*right panel*) maintains two allele classes and thereby leads to an elevated number of pairwise differences and average heterozygosity. *Bottom*: Frequency spectrum of segregating sites for the three cases. The number of chromosomes in which the rarer of two alleles is found is plotted on the x-axis (in this example, only $x = 1$ or $x = 2$ are possible); the y-axis shows the relative frequency among the total number of segregating sites (which is 4, 2, and 6 for the three cases, respectively).

11.6.2 Heterozygosity

Let us return to heterozygosity, H, the quantity we already discussed in the context of models that move forward in time (Chapter 10). Recall that H is the probability that two randomly drawn homologous sequences differ. As we move along the coalescent from the present into the past, the probability of a coalescence event is $1/(2N)$ in each generation, while the probability of a mutation is $1-(1-\mu)^2 \approx 2\mu$. Given either one event occurs, the probability that mutation occurs first, thereby creating a pair of distinct alleles, is the ratio

$$p_{\text{mut}} = \frac{2\mu}{2\mu + 1/(2N)} = \frac{\theta}{1+\theta}\,.$$

Thus, the random variable X with the two states $X = 0$ for "coalescence first" and $X = 1$ for "mutation first" has the distribution $P(X = 0) = 1/(1+\theta)$ and $P(X = 1) = \theta/(1+\theta)$. Given sample size n, p_{mut} is the same for any of $n(n-1)/2$ pairwise comparisons. For each of the pairwise comparisons there is a Bernoulli random variable X_i, $1 \le i \le n(n-1)/2$, with the two states $X_i = 0$ and $X_i = 1$. They are all identically distributed with parameter p_{mut}, but not independent. Average heterozygosity is the expected value of the random variable

$$H = \frac{2}{n(n-1)} \sum_i X_i.$$

Since

$$E\{H\} = \frac{2}{n(n-1)} \sum_i E\{X_i\},$$

average heterozygosity is identical to

$$E\{H\} = p_{\text{mut}} = \frac{\theta}{1+\theta}\,. \tag{11.10}$$

We already derived this result in Equation (10.21) using the recursion approach, but from the point of view of the coalescent we obtain an easy alternative derivation of this classical result [142].

Due to the lack of independence of X_i it is much harder to derive the variance $V\{H\}$ since is a function of sample size n. Considering the case $n = 2$, the variance is just the variance of the Bernoulli distribution

$$V_{n=2}\{H\} = \frac{\theta}{(1+\theta)^2}\,.$$

For large n the variance V_n is well approximated by the population variance of H. This has been obtained by Watterson [257] and is

$$V_{n=\infty}\{H\} = \frac{2\theta}{(1+\theta)^2(2+\theta)(3+\theta)}\,. \tag{11.11}$$

11.6.3 The Distribution of Segregating Sites

As a corollary to Watterson's result on expected heterozygosity (Eq. (11.10)) the entire distribution of the number of segregating sites, S, can be calculated explicitly for the case $n = 2$ [258] and in recursive form for general n. Averaging over coalescence times and for $n = 2$ one obtains

$$\begin{aligned}\text{Prob}_{n=2}(\bar{S} = i) &= \int_t \text{Prob}(S = i | T_2 = t) dP_{T_2} \\ &= \int_t \exp(-\mu t)\frac{(\mu t)^i}{i!} dP_{T_2} = \left(\frac{\theta}{1+\theta}\right)^i \frac{\theta}{1+\theta} \, .\end{aligned}$$

The recursion formula for arbitrary n is [120]

$$\text{Prob}_n(\bar{S} = i) = \sum_{j=0}^{i} \text{Prob}_{n-1}(i-j) Q_n(j) \, ,$$

where

$$Q_i(j) = \left(\frac{\theta}{\theta + i - 1}\right)^j \frac{i-1}{\theta + i - 1} \, .$$

We finally note that the distribution of segregating sites is well approximated by a Gamma distribution with parameters $\alpha = E^2\{S_n\}/V\{S_n\}$ and $\beta = V\{S_n\}/E\{S_n\}$ (Figure 11.10). These are uniquely determined by the mean and variance of the number of segregating sites (Eqs. (11.8) and (11.9)).

11.7 Simulation Example

One of the important features of the coalescent is that it can be implemented as a very efficient simulation tool. Using the coalescent simulation program `ms` (for **m**ake **s**ample) [122], it takes a few seconds to generate 100,000 gene samples of size $n = 20$. If we use $\theta = 10$ as input parameter, the expected number of SNPs according to Equation (11.8) is $S = \theta \sum_{i=1}^{19} 1/i = 35.48$. The simulated mean of $S = 35.52$ is virtually indistinguishable from this expectation (Fig. 11.10).

So far our evolutionary model only encompassed two events: coalescences and mutations. However, at least two other classes of events need to be incorporated to approach a tolerable semblance to evolution in nature: (i) recombination and (ii) selection, the latter being the central tenet of Darwin's theory of evolution [44].

11.8 Recombination

In the coalescent model with mutation, but without recombination, all nucleotides along a given chromosome have the same evolutionary history. However, in sexual eukaryotes recombination takes place in every generation during meiosis. In this

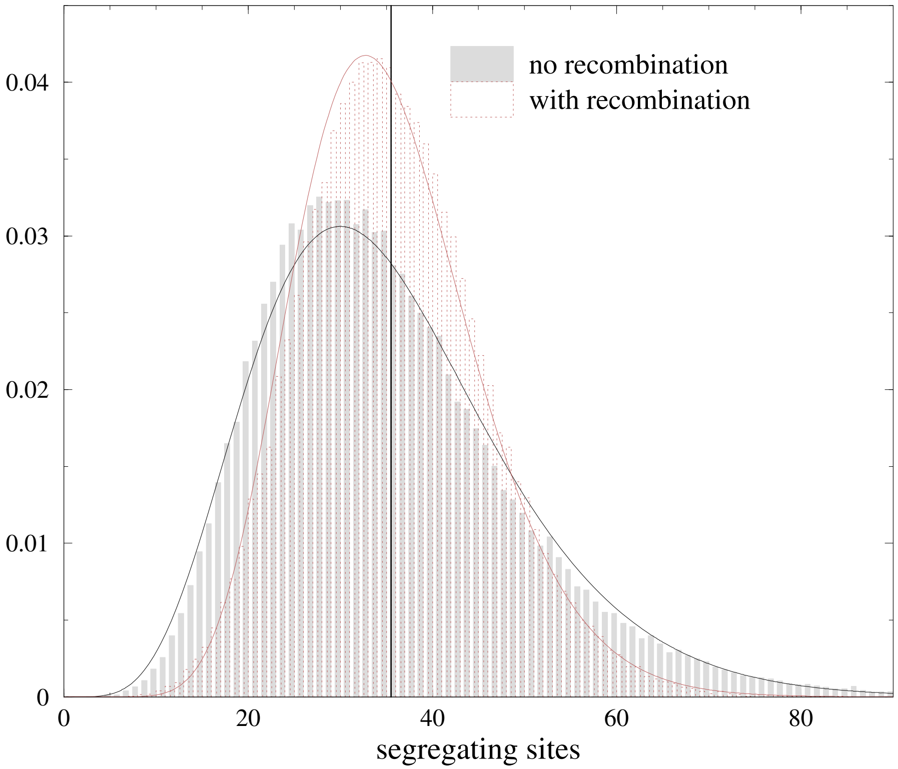


Fig. 11.10. The distribution of the number of segregating sites in 100,000 samples of size $n = 20$ with $\theta = 10$. *Solid histogram*: Neutral coalescent without recombination; *outlined histogram*: including recombination with rate $r = 10$. The black vertical line indicates the distribution's mean (35.52), which is almost indistinguishable from the expected mean (35.48). In both cases, with and without recombination, the means of the number of segregating sites are identical, although the distributions differ. The overlayed black and gray lines represent the densities of the approximating Gamma distribution. The simulation was carried out using the freely available program `ms` [122].

case, different chromosomal segments, e.g. a and b, may become unlinked and thus have distinct evolutionary histories. Let c be the rate per generation with which a recombination event occurs between the two segments in a given chromosome. This rate is proportional to the physical distance d between segments a and b on the chromosome. To a first order approximation one often assumes that $c = \rho \cdot d$, where ρ is the rate of recombination per nucleotide per generation. If $c = 0$, i.e. in the absence of recombination, the genealogies of a and b are identical. In contrast, if c becomes large, the two genealogies become independent of each other. When tracing the genealogy of two chromosomes back in time, a recombination event may precede the coalescent event (Fig. 11.11). In analogy to the case of mutation, the probability that a recombination event occurs first is

$$p_{\text{reco}} = \frac{2c}{2c + \frac{1}{2N}} = \frac{r}{1+r}\,,$$

where $r = 4Nc$.

As before, we wish to calculate the time to the most recent common ancestor, T_{MRCA}, and the number of segregating sites, S. For the two segments, a and b, there are two, usually distinct, most recent common ancestors (MRCAs), with corresponding times $T_{\text{MRCA}(a)}$ and $T_{\text{MRCA}(b)}$ (Fig. 11.11). For $n = 2$ the waiting time for one of the two events, recombination or coalescence, is geometrically distributed with mean $(1/(2N) + 2c)^{-1} = 2N/(1 + r)$. Notice that a recombination event increases the number of lineages by one (Fig. 11.11). As we follow the coalescent into the past, further recombination or coalescence events may take place. However, only those recombination events are relevant for the shape of the genealogy, which involve chromosomal segments occurring in the original sample. In contrast, recombination events involving the segments shown as dotted gray lines in Figure 11.11 would leave the genealogy unchanged, as they would not lead to the creation of new lineages. The generalization of the coalescent, which includes recombination, has also become known as the *ancestral recombination graph* and was introduced by Griffiths and Marjoram [92]. The number of ancestral nodes, i, at every point in time can be described as a birth and death process with rates $\lambda_b(i) = i \cdot r/2$ and $\lambda_d(i) = \binom{i}{2}$, respectively. The relative magnitude of these rates ensures the existence of a most recent common ancestor, although there is not only merging but also splitting of branches in a coalescent with recombination.

Recombination complicates the explicit derivation of central quantities such as T_{MRCA} and S. However, since recombination splits the coalescent into a number of subtrees, for each of these subtrees the expected total tree length remains identical to the case considered before. Therefore, in the presence of recombination we also have

$$E\{S\} = \theta \sum_{i=1}^{n-1} \frac{1}{i}\,.$$

Note that the argument is identical to the one which has been used above to derive expected heterozygosity (Eq. (11.10)): the expectation is a linear operator. And again for the same reason as above (the variance is not a linear operator), the variance turns out to be more complicated. Kaplan and Hudson [131] derived the following expression

$$V\{S\} \approx E\{S\} + \theta^2 V\{T\}\,, \tag{11.12}$$

where T is an average of the total sizes of the genealogies of all segments weighted by their lengths. Its variance is

$$V\{T\} \approx \frac{2\left(\sum_{i=1}^{n-1} \frac{1}{i^2}\right)}{r^2} \left(-r + \frac{23r + 101}{2\sqrt{97}} \log\left(\frac{2r + 13 - \sqrt{97}}{2r + 13 + \sqrt{97}}\;\frac{13 + \sqrt{97}}{13 - \sqrt{97}}\right) + \frac{r-5}{2} \log\left(\frac{r^2 + 13r + 18}{18}\right)\right).$$

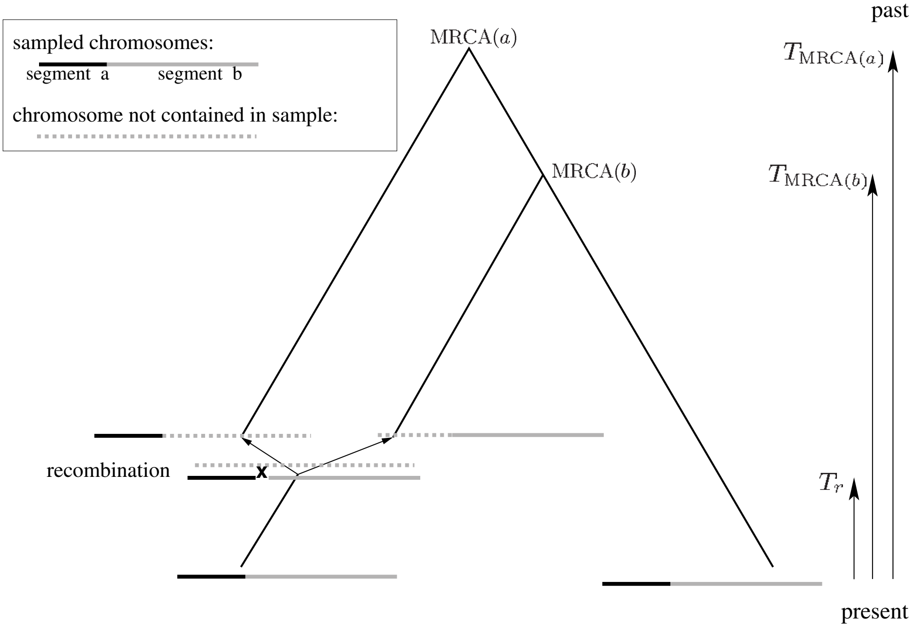


Fig. 11.11. Gene genealogy for sample size $n = 2$ with recombination separating segments a and b at time T_r.

In the limit of no recombination ($r \to 0$) $V\{T\}$ collapses to $\sum_{i=1}^{n-1} 1/i^2$ and as a consequence $V\{S\}$ turns into the expression known from Equation (11.9). Furthermore, $V\{T\}$ is monotonically decreasing in r and has the limit $\lim_{r\to\infty} V\{T\} = 0$. Therefore, for the variance of S we have $V_{r\to\infty}\{S\} = E\{S\}$. Thus, if all sites along a chromosome are uncoupled, i.e. evolve independently, the number of segregating sites is Poisson distributed. Linkage introduces deviation from the Poisson distribution, although the mutation process along the gene genealogy is still Poisson. However, in the presence of recombination the distribution of the number of segregating sites is still well approximated by a Gamma distribution (Fig. 11.10) with its location and scale parameters α and β uniquely determined by $E\{S\}$ and $V\{S\}$.

When considering only two segments, a and b as in the introductory example, an analytical expression for the correlation between times $T_{\mathrm{MRCA}(a)}$ and $T_{\mathrm{MRCA}(b)}$ has been found [131, 120]. It is

$$\mathrm{Cor}(T_{\mathrm{MRCA}(a)}, T_{\mathrm{MRCA}(b)}) = \frac{r+18}{r^2+13r+18}\,. \tag{11.13}$$

It confirms that in the absence of recombination $T_{\mathrm{MRCA}(a)}$ and $T_{\mathrm{MRCA}(b)}$ are completely correlated. Even stronger, $\mathrm{Cor}(T_{\mathrm{MRCA}(a)}, T_{\mathrm{MRCA}(b)}) = 1$ if and only if $r = 0$.

Notice that the coalescent with recombination is a kind of super-genealogy containing the genealogies of the constituent fragments, in our case a and b. In Figure 11.11 the a genealogy is longer than the b genealogy. We therefore expect segment a to contain a higher SNP density than segment b. Genealogies between neighboring segments are uncoupled by recombination and therefore the times to the MRCA for different segments may vary strongly. In apparent contradiction to this observation is the finding that recombination *reduces* the variance (see Fig. 11.10 and Eq. (11.12)) of the number of segregating sites and thus reduces the variance in genetic variability. However, while recombination introduces additional variation of coalescence times on a local scale, it has an averaging effect on genetic variability on a global scale.

11.9 Selection

We have seen that recombination can lead to drastic changes in the coalescent topology. This is also true for selection. In contrast to recombination, selection does not change the branching structure of the coalescent, but it changes the branch lengths. Depending on the mode of selection, time intervals between coalescence events can shrink or expand. Consider a mutation that confers a selective advantage. This mutation and its host chromosome have an increased probability (see Eq. (10.32)) of rising in the population, and may finally become fixed. Given that the mutation does become fixed, its rapid spread through the population, once it has reached a moderate frequency, compresses the coalescent, leading to what is often referred to as a *star-like phylogeny*. In Figure 11.12 six of the seven coalescence events take place during the spread of the advantageous allele. In contrast, there is only one coalescence event during the subsequent neural phase of evolution. One effect of the star shape of the coalescent with selection is that neutral mutations appear mainly on the outer branches of the genealogy. This implies an excess of singletons or *rare* polymorphisms compared to what would be expected under neutral evolution.

The effects of different types of selection on the shape of the coalescent and, hence, on the distribution of SNPs is illustrated in Figure 11.9. The neutral genealogy shown on the left panel forms the background to all discussions of selection. The kind of selection that favors one allele over another is known as *directional selection*. The majority of non-neutral mutations are harmful to an organism and are therefore selected against. This type of directional selection is called negative, or purifying, selection. The recurrent removal of disadvantageous mutants from a population is termed *background selection* [38]. Occassionally, a non-neutral mutant may provide a fitness advantage, for instance when it renders its carrier better adapted to given environmental conditions. This type is called positive, or adaptive, selection. The effect on the shape of the coalescent of such an adaptive mutation in the absence of recombination is shown in the middle panel of Figure 11.9. It causes a star-like phylogeny (cf. Fig. 11.12) that leads to an excess of singleton mutations.

A different scenario is created by the existence in the population of two alleles that are beneficial only if present at the same time. A standard example for such genetic synergism is sickle-cell anemia, where the combined presence of the sickle allele of β-globin and its wild type confers heightened malaria resistance [106, p.

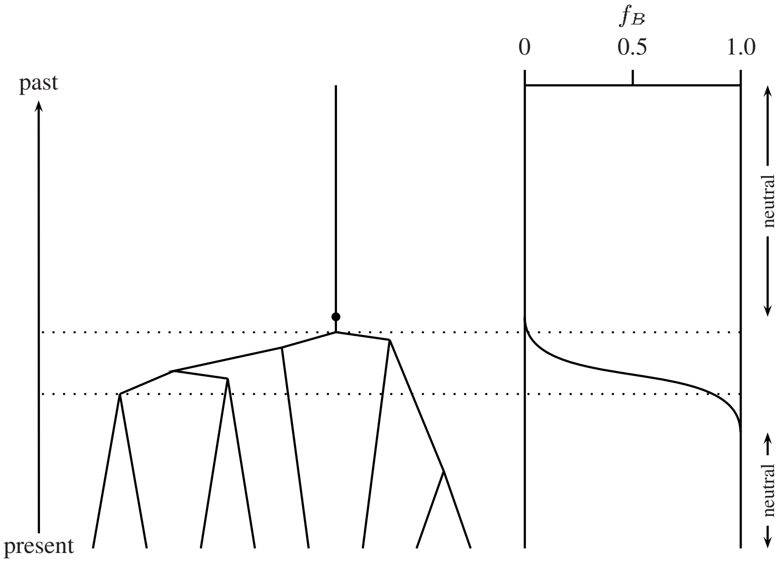


Fig. 11.12. Genealogy with selection (sample size $n = 8$). A positively selected mutation, B, (dot) gets fixed in the population and thereby dominates the genealogy of the sample in the present. The frequency of the selected allele, f_B, is plotted on the righthand side. The dotted lines indicate the time interval with an elevated rate of coalescence events while allele B rises quickly to fixation, also called a "selective sweep". The sweep is flanked by two phases of neutral evolution.

230]. The resulting *balancing selection* leads to deep branching in the coalescent and correspondingly a bimodal distribution of pairwise differences as shown in Figure 11.9 (right panel).

In summary, for a given sample of homologous DNA sequences drawn from a panmictic population, the number of SNPs can easily be established. Coalescent theory makes predictions about distributional properties of the number of segregating sites under diverse evolutionary scenarios. These theoretical predictions can be compared to experimental observations. The number of evolutionary processes that can be incorporated in coalescent models is only limited by a researcher's ability to implement the corresponding software. The widely used program `ms` incorporates reciprocal recombination as well as gene conversion, migration between subpopulations, and changes in population size over time [122].

11.10 Combining Recombination and Selection

Coalescent models that combine selection and recombination can be used to determine the influence of a selected allele on the number and distribution of polymorphisms in its chromosomal neighborhood. With low rates of recombination, the fate of this neighborhood is tightly coupled to the selection dynamics (see Fig. 11.9), whereas with large recombination rates it is essentially unaffected by the kind of selection upon neighboring alleles.

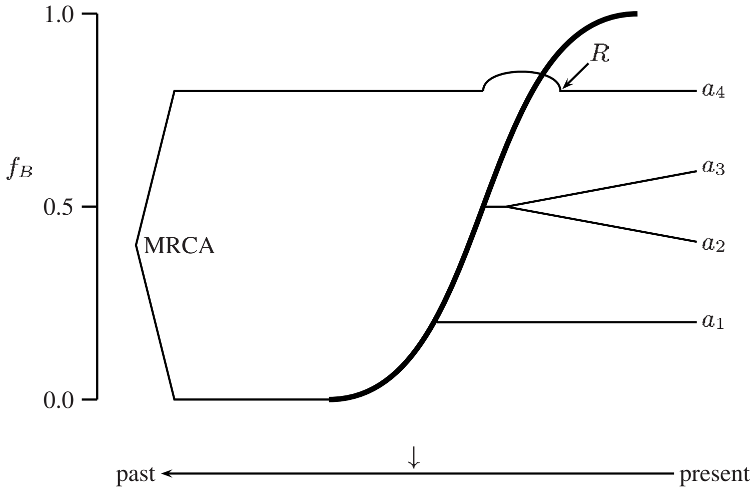


Fig. 11.13. "Hitchhiking" with a selected allele. The genealogy of the neutral alleles a_1,...,a_4 depends on whether or not recombination takes place during the fixation of allele B. Without the recombination event (R), the MRCA of the alleles in the sample would be much younger, for example at time ↓, because in that case all a alleles would be descendants of a B-carrying chromosome. The frequency of allele B, f_B, is indicated by the bold line.

We now concentrate on the interaction of recombination and positive directional selection. In this case, the neighborhood of the selected allele can be regarded as hitching a ride to fixation with the adaptive trait. Hence, this evolutionary scenario is also known as *genetic hitchhiking* [174, 227]. The effect of hitchhiking depends strongly on recombination during the selected allele's rise to fixation as illustrated in Figure 11.13. Recombination leads to an uncoupling between a selected allele and other portions of the chromosome. Conversely, in the absence of recombination any episode of directional selection at any gene along a chromosome would impose a star phylogeny on all these genes. Such a star phylogeny is characterized by low levels of diversity (cf. Fig. 11.9, directional selection). This is the reason why it is difficult to reject neutrality on the basis of finding the supposedly low genetic diversity of $H \approx 0.2$ in *E. coli* [182] as discussed in Section 10.6. This bacterium reproduces asexually and has essentially a clonal population structure [222]. Due to periodic selection such a population structure leads to a genome-wide reduction in genetic diversity even in very large populations. How strong the effect of periodic positive selection is on the levels of genetic diversity depends on its frequency and intensity [164].

In the presence of recombination genetic diversity may also be reduced due to recurrent episodes of positive selection. However, the effect depends additionally on

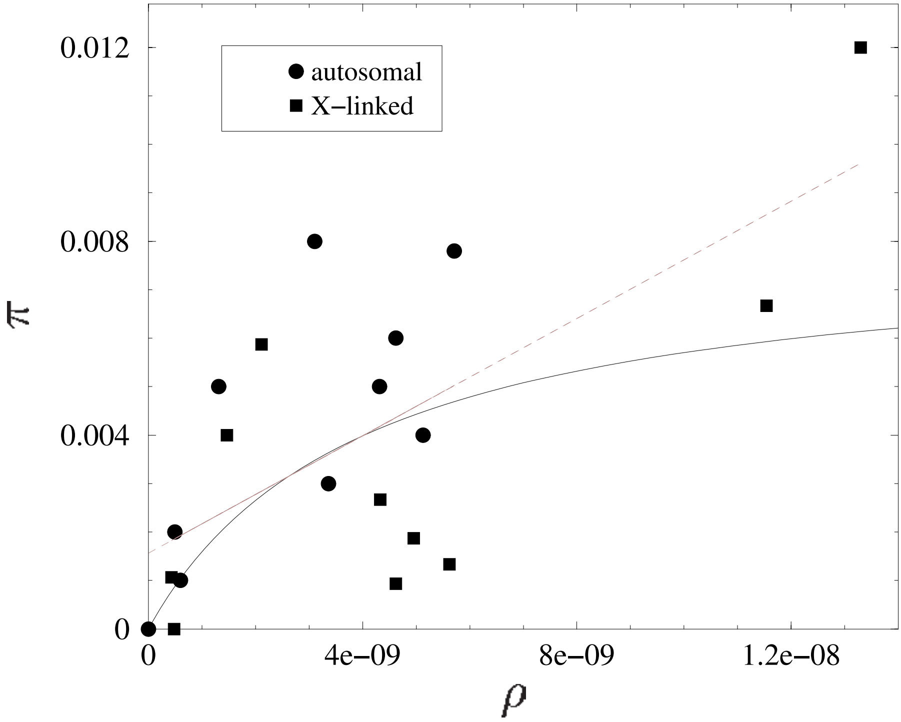


Fig. 11.14. Scatter plot of genetic diversity per bp (π) versus recombination rate (ρ) for twenty loci from *Drosophila melanogaster* (Table 11.2). Genetic diversity tends to be lower in regions of low recombination. Linear regression analysis yields a correlation coefficient of $+0.68$ for these data (straight line). The curved line is obtained by fitting the parameters $\alpha = 2Ns$, k_s and π_{neut} of the equation $\pi = (\pi_{\text{neut}}\,\rho)/(\rho + k_s \cdot f(\alpha))$ to the data [260]. The quantity π_{neut} is the equilibrium level of genetic diversity per bp under neutral evolution. Population size was estimated independently ($\hat{N} = 2 \cdot 10^6$). For the data shown, the estimates are $\hat{\pi}_{\text{neut}} = 0.0080$, $\hat{k}_s = 5.3 \cdot 10^{-13}$ and $\hat{s} = 0.03$.

the magnitude of the regional recombination rates [132, 236, 20]. In fact, as a result of recurrent positive selection, one expects to observe a positive correlation between the recombination rate and genetic diversity. Figure 11.14 shows measurements of recombination rates and genetic diversity which were collected for a set of loci in *Drosophila melanogaster* (Table 11.2). Genetic diversity refers here to average heterozygosity per bp, which is commonly abbreviated by the letter π. The positive slope of the regression line shown in Figure 11.14 confirms this expectation. Given that selection was the driving agent to produce the positive correlation of ρ and π and given a model of the hitchhiking effect, more information can be extracted from such data. In one model [260] an explicit relationship between genetic diversity and the recombination rate has been derived, where quantities such as the selection coefficient, s, and the rate of selected substitutions, k_s, are contained as parameters. Fitting

Table 11.2. Measurements[1] of recombination rate per bp (ρ) and genetic diversity per bp (π) for twenty gene loci in *Drosophila melanogaster*.

locus	$\rho \cdot 10^{-9}$	π	reference
X-linked loci			
Yellow,Achaete	0.429	0.0008	[19]
Pgd	1.466	0.0030	[19]
Z,Tko	2.114	0.0044	[4]
Per	4.952	0.0014	[19]
White	13.30	0.0090	[183]
Notch	11.54	0.0050	[216]
Vermilion	5.619	0.0010	[20]
Forked	4.330	0.0020	[160]
Zw	4.619	0.0007	[53]
Su(F)	0.476	0.0000	[160]
autosomal loci			
Gpdh	5.714	0.0078	[241]
Adh	4.621	0.0060	[161]
Ddc	1.314	0.0050	[20]
Amy	3.107	0.0080	[162]
Pu	5.129	0.0040	[241]
Est6	4.314	0.0050	[84]
MtnA	0.593	0.0010	[159]
Hsp70A	0.493	0.0020	[163]
Ry	3.364	0.0030	[12]
Ci^D	0.000	0.0000	[21]

[1] In order to compare autosomal and X-linked data they need to be standardized. Since in *Drosophila* recombination takes place only in female flies, ρ has to be multiplied by $1/2$ for autosomal loci and by $2/3$ for X-linked loci. To compensate for the differences in effective population sizes for X-linked and autosomal loci, π from X-linked loci needs to be multiplied by $4/3$. The standardized data are plotted in Figure 11.14.

these parameters to the experimental data, the curved graph of π versus ρ shown in Figure 11.14, emerges. A parameter fitting procedure yielded estimates of $\hat{s} = 0.03$ and $\hat{k}_s = 5.3 \cdot 10^{-13}$ per nucleotide per generation for these data [260]. That is, on average a selected allele has a three percent better chance of being passed on to the next generation than its neutral homologue. Such analyses shed light on the old question regarding the selective value of molecular polymorphisms. In contrast to the early phase of the selectionists *vs.* neutralists controversy, this debate is today informed by a wealth of molecular data on sequence polymorphisms seen within species and populations. Based on these data, hitchhiking models are now widely used to trace individual selective sweeps in a genome in addition to quantifying an overall rate of positively selected substitutions. This area of research, with the scope of identifying those sites in a genome which currently are, or recently have been, under adaptive evolution, is known as *hitchhiking mapping* [103, 217, 188]. Together with a set of tests for neutrality, which is the topic of Chapter 12, the statistical significance of putatively non-neutral regions in a genome can be established.

11.11 Summary

In this chapter we have turned the Wright-Fisher model of gene dynamics on its head. Given a sample of n genes, their lineages coalesce into common ancestors as we move backward in time. If we trace back the gene genealogies until the most recent common ancestor of the entire sample, this sample genealogy is called the coalescent. Coalescent theory allows the derivation of important evolutionary quantities such as the number of segregating sites expected under neutrality. In addition, the coalescent is routinely used as a very efficient tool for simulating gene samples under a diverse range of evolutionary scenarios, including recombination and selection.

11.12 Further Reading

A classical review of coalescent theory is written by Hudson, one of the founders of coalescent theory [120]. More recently a number of surveys of this approach to evolutionary modeling have been published, the most accessible of which is perhaps an article by Nordborg [192]. In addition, Hein, Shierup, and Wiuf provide an introductory monograph on coalescent theory [112].

11.13 Software Demonstrations and Exercises

11.1. The `bioinformer` software contains under `Evolution` → `Coalescent` a program that visualizes neutral coalescent trees with mutation and the corresponding SNP patterns. Start this program and work through the corresponding tutorial in Section A.4.4.

11.2. The `bioinformer` also contains under `Evolution` → `WrightFisher` a program that visualizes ancestral lineages in a Wright-Fisher population. Work through the tutorial for that program in Section A.4.3.

11.3. How many SNPs do you expect under the neutral infinite sites model in a sample of $n = 10$ genes given $\theta = 4N\mu = 10$?
What is the sample size needed in a haploid population of size $N = 50$ to get an average time to the most recent common ancestor of 10 generations?

11.4. Use the freely available coalescent program `ms` [122] to simulate 10,000 gene samples given the parameters $n = 20$ and $\theta = 10$.

1. How long does the simulation take?
2. What is the mean and variance of the number of SNPs in the simulated samples?

11.5. Repeat the simulation, but this time include reciprocal recombination with $r = 10$ and $\theta = 10$.

1. How long does the simulation take?
2. What is the mean and variance of the number of SNPs in the simulated samples?
3. Compare your results to the simulations without recombination.

11.6. In coalescent genealogies with recombination there is splitting and merging of lineages. Do all coalescent genealogies with recombination have a most recent common ancestor?

12

Testing Evolutionary Hypotheses

> *For many years population genetics was an immensely rich and powerful theory with virtually no suitable facts on which to operate.*
>
> *Richard C. Lewontin [166, p. 189]*

Experimental data such as the eleven *Adh* sequences of *D. melanogaster* summarized in Table 10.1 represent a single realization of an evolutionary process. In biology, such data cannot be arbitrarily replicated since—in most cases—the evolution experiment conducted by nature over long periods of time cannot readily be repeated. This begs the question of how data of a single realization of a stochastic process can be evaluated and interpreted. The answer is given by a number of statistical tests, which are designed to compare scarce data with a probability distribution and to ask whether or not the data are extreme with respect to the distribution. Such probability distributions can sometimes be obtained analytically and one example of this is Ewens' sampling formula introduced in Section 10.7. However, in most cases they are generated by computer simulations of the evolutionary process. Out of the multitude of methods that have been developed over the years to test various evolutionary hypotheses, we are going to concentrate in this chapter on examples from two areas: tests of the neutral mutation hypothesis and tests of recombination.

12.1 Hudson-Kreitman-Aguadé (HKA) Test

According to the neutral mutation hypothesis, the genetic variability between populations should be correlated with the variability within populations. The neutrality test by Hudson, Kreitman, and Aguadé, also known as the HKA test, assesses this prediction [124]. It requires sequence data from $L \geq 2$ loci in two species that form the basis for three statistics:

1. S_i^A: number of segregating sites at locus i in species A, i.e. the within-species diversity at locus i;
2. S_i^B: number of segregating sites at locus i in species B;
3. D_i: number of nucleotides at which two randomly selected sequences at locus i from species A and B differ, i.e. the between-species divergence at locus i.

The test is based on the following assumptions:

1. discrete generations;
2. mutations are neutral, independent, Poisson-distributed, and non-recurrent (neutral infinite sites model);
3. there is no recombination within loci;
4. all loci are unlinked, i.e. in linkage equilibrium;
5. species A and B have reached a constant size of $2N$, and $2Nf$, respectively;
6. species A and B have diverged from a common ancestral population T' generations ago; the size of this ancestral population was the average of the two species' present population size, i.e. $2N(1+f)/2$.

Given this set of rather conservative assumptions, expectations (E) and variances (Var) for the central quantities of the test can be derived:

$$\begin{aligned} E\left(S_i^{\mathrm{A}}\right) &= \theta_i H_{n_{\mathrm{A}}-1}, \\ \mathrm{Var}\left(S_i^{\mathrm{A}}\right) &= E\left(S_i^{\mathrm{A}}\right) + \theta_i^2 H_{n_{\mathrm{A}}-1}^{(2)}, \\ E\left(D_i\right) &= \theta_i\left(T + (1+f)/2\right), \\ \mathrm{Var}\left(\theta_i\right) &= E\left(D_i\right) + \left(\theta_i(1+f)/2\right)^2, \end{aligned} \tag{12.1}$$

where $1 \leq i \leq L$, n_{A} is the size of the sample drawn from species A, $\theta_i = 4N\mu_i$ the neutral mutation parameter of a diploid population, N the population size, μ_i the number of mutations per generation at locus i, $T = T'/2N$, H_n the harmonic number,

$$H_n = \sum_{i=1}^{n} \frac{1}{i},$$

and $H_n^{(r)}$ the harmonic number of order r:

$$H_n^{(r)} = \sum_{i=1}^{n} \frac{1}{i^r}.$$

The system of Equations (12.1) contains the unknown quantities θ_i, f, and T. Their estimators, $\hat{\theta}_i$, $\hat{f}$, and $\hat{T}$, were obtained by solving the following system of equations:

$$\begin{aligned} \sum_{i=1}^{L} S_i^{\mathrm{A}} &= H_{n_{\mathrm{A}}} \sum_{i=1}^{L} \hat{\theta}_i, \\ \sum_{i=1}^{L} S_i^{\mathrm{B}} &= \hat{f} H_{n_{\mathrm{B}}} \sum_{i=1}^{L} \hat{\theta}_i, \\ \sum_{j=1}^{L} D_i &= \left(\hat{T} + (1+\hat{f})/2\right) \sum_{i=1}^{L} \hat{\theta}_i, \\ S_i^A + S_i^B + D_i &= \hat{\theta}_i\left(\hat{T} + (1+\hat{f})/2 + H_{n_{\mathrm{A}}} + \hat{f} H_{n_{\mathrm{B}}}\right), \\ & 1 \leq i \leq L-1. \end{aligned} \tag{12.2}$$

By replacing the parameters in Equations (12.1) with their estimators, we get estimators for the central quantities used in the test. Assuming that the within population diversity and the between population divergence are normally distributed, the following test statistic is approximately χ^2 distributed [231]:

$$X^2 = \sum_{i=1}^{L} \frac{\left(S_i^{\mathrm{A}} - \hat{E}\left(S_i^{\mathrm{A}}\right)\right)^2}{\hat{\mathrm{V}}\mathrm{ar}\left(S_i^{\mathrm{A}}\right)} + \sum_{i=1}^{L} \frac{\left(S_i^{\mathrm{B}} - \hat{E}\left(S_i^{\mathrm{B}}\right)\right)^2}{\hat{\mathrm{V}}\mathrm{ar}\left(S_i^{\mathrm{B}}\right)} + \sum_{j=1}^{L} \frac{\left(D_i - \hat{E}\left(D_i\right)\right)^2}{\hat{\mathrm{V}}\mathrm{ar}\left(D_i\right)}.$$

To see how this works, let us take a look at the application given by the authors of the HKA test [124]. Table 12.1 summarizes their diversity data from two fruitfly species, *D. melanogaster* and *D. sechellia*. The data were obtained using restriction length polymorphisms. Since restriction enzymes only sample a subset of the positions in a target sequence, Table 12.1 draws a distinction between the lengths of the sequences surveyed and the number of sites compared. Moreover, it is clear that the *Adh* locus and its 5' flanking region are linked, thereby violating one of the assumptions of the test. However, the authors showed that the test is conservative with respect to its assumption of no linkage between loci. That is, if the loci tested are linked, it becomes more difficult to reject the null hypothesis of neutrality and the same applies to the assumption of no intra-genic recombination [124].

Table 12.1. Restriction fragment length polymorphism (RFLP) data to illustrate the HKA test: distribution of variable sites (Var. sites) around the alcohol dehydrogenase (*Adh*) locus in *D. melanogaster* and between *D. melanogaster* and *D. sechellia*. Data taken from [124].

	5' Flanking			*Adh* locus		
	Length	Sites	Var. sites	Length	Sites	Var. sites
Within species ($n = 81$)	4000	414	9	900	79	8
Between species	4052	4052	210	900	324	18

Let θ' refer to the per *nucleotide* neutral mutation parameter, in contrast to the θ used so far, which refers to the per *locus* neutral mutation parameter. Further, we identify species A with *D. melanogaster* and species B with *D. sechellia*. By substituting the data on the distribution of genetic variation from Table 12.1, the top equation in system (12.1) describing the within-species diversity becomes

$$S_1^A + S_2^A = 9 + 8 = \left(414\hat{\theta}_1' + 79\hat{\theta}_2'\right) H_{80}. \tag{12.3}$$

There was no data available on within-species diversity. As a substitute it was simply assumed that the ancestral population of species B had the same size as that of species A, i.e. that in the third and fourth equations of system (12.1) $f = 1$. The remaining equations of system (12.1) describing between-species diversity and connecting within- and between-species diversity therefore become

$$\begin{aligned} D_1 + D_2 &= 210 + 18 \\ &= \left(4052\hat{\theta}'_1 + 324\hat{\theta}'_2\right)\left(\hat{T} + 1\right), \\ D_1 + S_1^A &= 210 + 9 \\ &= 4052\hat{\theta}'_1\left(\hat{T} + 1\right) + 414\hat{\theta}H_{80}. \end{aligned} \tag{12.4}$$

Solving Equation (12.3) and system (12.4) yields $\hat{T} = 6.73$, $\hat{\theta}'_1 = 6.6 \cdot 10^{-3}$, $\hat{\theta}'_2 = 9.0 \cdot 10^{-3}$, and $X^2 = 8.15$. This X^2-value corresponds to a P-value of 0.004, assuming a χ^2 distribution with 1 degree of freedom; in other words, the data did not agree with the neutral mutation hypothesis.

In order to discover the kind of evolutionary force that might have caused this rejection of the neutral mutation hypothesis, it is necessary to reconsider the experimental data (Table 12.1). There was an approximately equal amount of divergence between species ($210/4052 = 0.052$ and $18/324 = 0.056$). In contrast, the diversity within species was approximately four times higher at the *Adh* locus than in the flanking region ($8/79 = 0.101$ *vs.* $9/414 = 0.022$). The authors therefore conclude that balancing selection is likely to affect the coding region of the *Adh* locus in fruit flies [124].

12.2 Tajima's Test

As we have seen, the HKA test requires polymorphism data from two species. In contrast, Tajima's test of the neutral mutation hypothesis is based on a sample of sequences drawn from a single population [240]. Consider an alignment of n homologous sequences. For each of the possible $\binom{n}{2}$ pairs of sequences we can count the number of positions at which they differ. The arithmetic average of this number is defined as Π. This is an estimator of the neutral mutation parameter $\theta = 4N_e\mu$ and we define

$$\theta_\Pi = \Pi.$$

Like the average number of pairwise differences, the number of segregating sites is also a function of θ as well as of the sample size n: $\theta = S/H_{n-1}$. We therefore define a second estimator of θ:

$$\theta_S = \frac{S}{H_{n-1}}.$$

Based on these two estimators of θ, Tajima suggested the test statistic

$$D = \frac{\theta_\Pi - \theta_S}{\sqrt{\mathrm{Var}\left(\theta_\Pi - \theta_S\right)}}. \tag{12.5}$$

Under strict neutrality $\theta_\Pi = \theta_S$ and, hence, $D = 0$. The denominator ensures that the standard deviation of D is 1.

To understand how D behaves under non-neutrality, notice that as long as the number of alleles in a sample is constant, S is independent of allele frequencies. This leads to a disproportionately large contribution of rare alleles to S. Since deleterious alleles are rare, such alleles lead to a situation where $\theta_\Pi < \theta_S$. $D < 0$ is therefore indicative of purifying selection. Alternatively, if a gene is fixed through rapid selection of an advantageous allele, neutral genetic diversity linked to the advantageous variant is also swept to fixation. Such a phenomenon is known as a "selective sweep" or "genetic hitchhiking" [174] and is characterized by an excess of low-frequency alleles in the sample (cf. Section 11.10). This also leads to an inflation of θ_S relative to θ_Π and, hence, to a negative value of Tajima's D. Conversely, alleles that grow from low to intermediate frequencies increase Π while leaving S unchanged. A value of $D > 0$ therefore indicates balancing selection.

In order to actually apply Equation (12.5), we still need to estimate the statistic's denominator. This is obtained from equations given by Tajima [240]:

$$\hat{\mathrm{V}}\mathrm{ar}\,(\theta_\Pi - \theta_S) = \frac{S(3 - 3n + (n+1)H_{n-1})}{3(n-1)H_{n-1}^2} + \frac{(S-1)S\left(\frac{2(3+n+n^2)}{9n(n-1)} - \frac{2+n}{nH_{n-1}} + \frac{H_{n-1}^{(2)}}{H_{n-1}^2}\right)}{H_{n-1}^2 + H_{n-1}^{(2)}}.$$

As an example application for Tajima's D we return to the *FOXP2* gene introduced in Section 9.8.1. A single amino acid replacement in the DNA-binding domain of the FOXP2 protein causes severe articulatory difficulties as well as linguistic and grammatical impairment in humans [63]. In a survey of this gene, 14,063 nucleotides were sequenced from 40 human *FOXP2* genes. Table 12.2 shows the 47 polymorphic sites detected, which occurred in 17 unique haplotypes. The dominant haplotype accounted for over half of the sample (21 copies), three other haplotypes occurred twice, and the remaining 13 were unique [63]. Figure 12.1 shows the frequency and distribution of the 47 SNPs along the region surveyed. Two-thirds (31) of the polymorphisms occurred just once, which means they were generated by mutations that affected either a single, or, though less likely, 39 genes in the sample. This excess of rare variants is reflected in the inflated value of $\theta_S = 11.06$ compared to $\theta_\Pi = 4.22$ and a correspondingly small $D = -2.20$ [63]. This is certainly less than zero, but is this deviation from neutral expectation significant?

There are two methods for assessing the statistical significance of a given value of D: by assuming that D follows a known null distribution, or by simulating this null distribution. In the original publication of the test, Tajima suggested that the null distribution of D is approximated by the beta distribution. However, the application of assumed null distributions for D has been criticized [81] and coalescent simulations are often used instead. The results of such simulations for the *FOXP2* data are shown in Figure 12.2. The neutral coalescent was used to generate 100,000 artificial data sets conditioned on the sample size and the observed genetic diversity. From

Table 12.2. The 17 unique haplotypes detected in a survey of 14,063 nucleotides at 40 human *FOXP2* loci. A dot indicates a match to the nucleotide in the top line. *Count* gives the number of copies of the respecitve haplotype. Adapted from [63].

#	Count	Haplotype
1	21	AAATAGAAACACCAGAGAGAGACGTCCGATACGTGCTATAATGATCG
2	2	..............A....................T...........
3	2	..G..
4	2	...G..........A.......T..TA..A..........G...C.A
5	1	.......................A.......................
6	1	..............A................................
7	1	..............A.........C..A..............A....
8	1	.......................A.....................T.
9	1	C..G.AGT...TTGA....TA.......CA..CC..A.GT...G...
10	1	..............................G................
11	1	........G.....A......G.............T...........
12	1	........G.....A....................T...........
13	1	...G..........A.......T..TA..A.T........G...C.A
14	1	....G..........G...............................
15	1	...G......G...A.AGC......TA..A....TT.....C.....
16	1	.T...
17	1	.........T....A...C...............TT.C...C.....

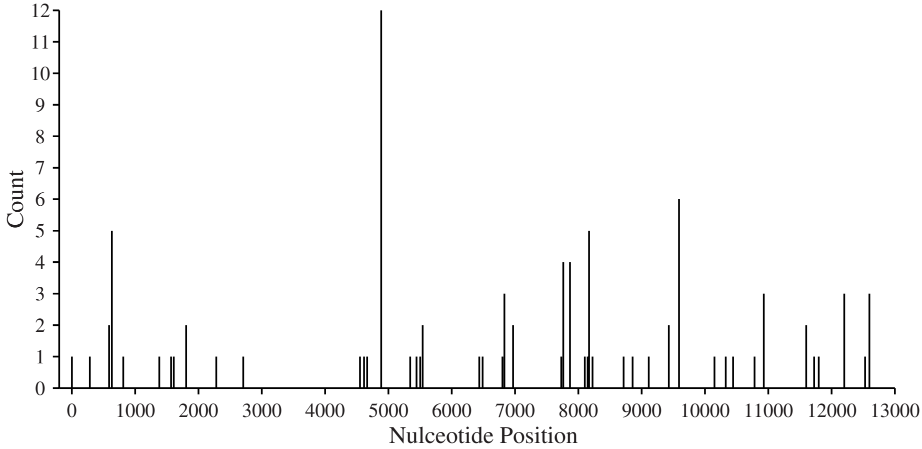


Fig. 12.1. Position and count of the rare allele at each single nucleotide polymorphism (SNP) along the 14,063 nucleotides surveyed at 40 human *FOXP2* loci. The position of the first SNP is set to 1. Adapted from [63].

each of these data sets D was calculated and the resulting null distribution of D has a mean close to 0 (-0.06), as expected. The observed value of $D = -2.20$ implies that $P = 0.0019$, which agrees with the hypothesis that *FOXP2* was affected by a recent selective sweep [63].

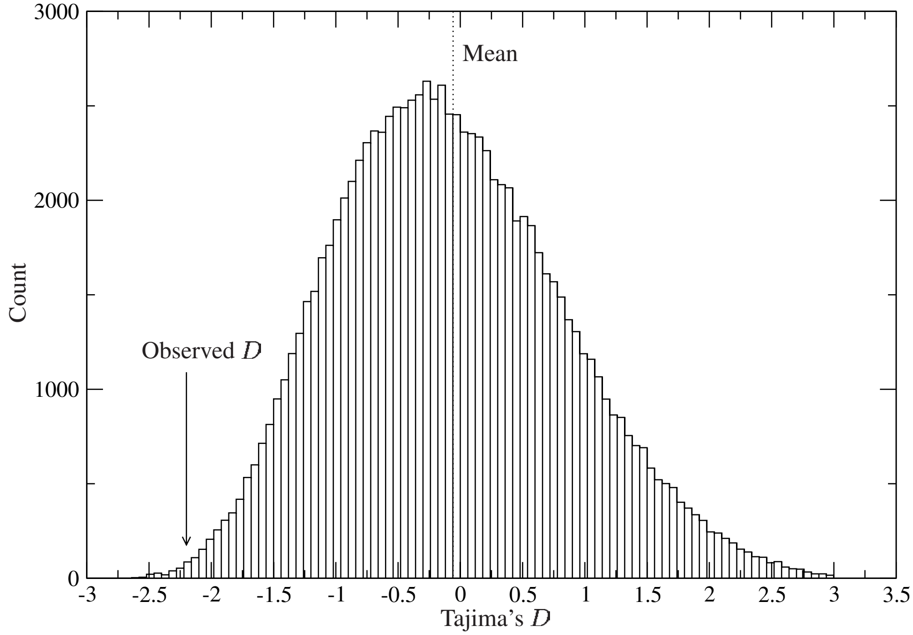


Fig. 12.2. Testing neutrality at the *FOXP2* locus in humans. Distribution of 100,000 values of Tajima's D calculated using the neutral coalescent with $n = 40$ and $\theta = 4.22$, as observed in the *FOXP2* sample shown in Table 12.2. The position of the arrow indicates that under a neutral model of evolution the observed value of D is highly unlikely ($P = 0.0019$). Coalescent simulations were carried out using the software `ms` [122].

12.3 Fu and Li's Test

The test of neutrality due to Fu and Li is similar in spirit to Tajima's test statistic, but its justification is based squarely on coalescent considerations. The central idea of the test is that under neutrality the number of mutations on external branches of a genealogy is equal to the number of mutations on its internal branches. Since deleterious mutations tend to have arisen recently in a sample's evolutionary history, they should accumulate on the external branches of the coalescent. As illustrated in Figure 12.3A, such evolutionary events result in singleton mutations. The converse, however, is not always true. In order to see that not every singleton has necessarily arisen on the external branches of a coalescent, consider the genealogy shown in Figure 12.3B. Any mutation along branch a will cause a SNP pattern that is indistinguishable from a singleton mutation. This constellation is not possible, if neither of the two branches leading to the genealogy's root is external, as is the case in Figure 12.3A. Taking this into account, Fu and Li suggested the following test statistic:

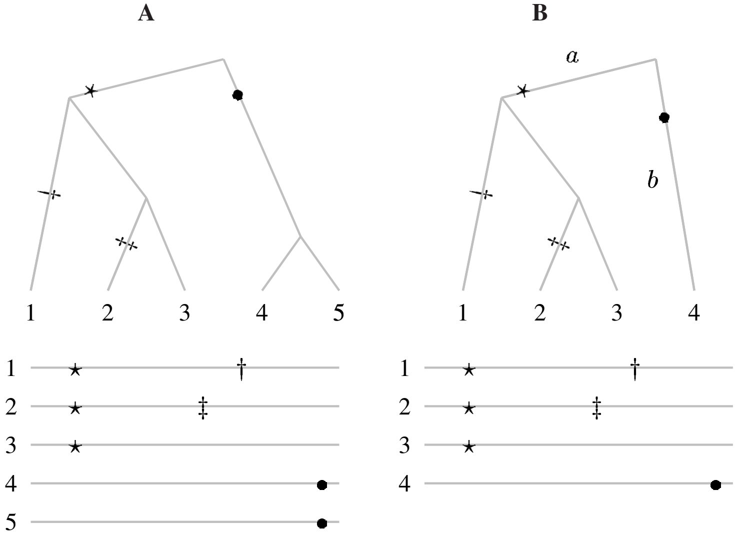


Fig. 12.3. Relationship between the topology of a gene genealogy, the distribution of mutations ($\star, \dagger, \ddagger, \bullet$) on this genealogy, and the resulting pattern of single nucleotide polymorphisms (SNPs). **A:** On a topology where neither of the two branches joined at the root are external, all singleton mutations ($\dagger, \ddagger$) must have arisen on one of the genealogy's external branches. **B:** If one of the two branches leading to the root is external, mutations arising in the two branches a and b both result in singleton SNPs.

$$G = \frac{\frac{n}{n-1}S - H_{n-1}S_{\mathrm{s}}}{\sqrt{\mathrm{Var}\left(\frac{n}{n-1}S - H_{n-1}S_{\mathrm{s}}\right)}}, \tag{12.6}$$

where S_{s} is the number of singleton mutations. The variance of the numerator of Equation (12.6) is obtained from the equations given by Fu and Li [81] as

$$\mathrm{Var}\left(\frac{n}{n-1}S - H_{n-1}S_{\mathrm{s}}\right) = S\left(H_{n-1} - 2 - A\right) + S^2\left(1 + A\right),$$

where

$$A = \frac{H_{n-1}^2 \left(\frac{-2(n-1)+2nH_{n-1}}{(n-2)(n-1)} - \frac{n+1}{n-1} \right)}{H_{n-1}^2 + H_{n-1}^{(2)}}.$$

The *FOXP2* data set we have already investigated using Tajima's D (Table 12.2) contains $n = 40$ sequences and $S = 47$ polymorphisms, of which $S_{\mathrm{s}} = 31$ are singletons. Substitution of these numbers into Equation (12.6) yields $G = -3.50$, which is significant according to the critical values provided by Fu and Li ($P < 0.01$) [81]. This is in good agreement with the result we obtained using Tajima's D.

12.4 McDonald-Kreitman Test

Like the HKA test, the McDonald-Kreitman test is based on sequence data from at least two species [178]. However, in this case a single locus suffices, which needs to be protein-coding. First, all nucleotide positions in such a data set are classified as *fixed differences* or *polymorphisms*. A position represents a fixed difference, if it is monomorphic within but polymorphic between the two species investigated (Fig. 10.3B). A position is polymorphic, if it is variable within either or both of the species. Next, the fixed and polymorphic positions are classified into synonymous and non-synonymous substitutions. Under the neutral mutation hypothesis, sequence changes accumulate at a constant rate along branches, regardless of whether they lead to species or to individuals within species. This implies that the ratio of synonymous to non-synonymous substitutions should be equal between fixed and polymorphic positions. That is,

$$H_0 : d_{\mathrm{nf}}/d_{\mathrm{sf}} = d_{\mathrm{np}}/d_{\mathrm{sp}},$$

where d_{nf} and d_{sf} refer to the numbers of non-synonymous and synonymous fixed differences, respectively. Similarly, d_{np} and d_{sp} denote the number of non-synonymous and synonymous polymorphisms.

McDonald and Kreitman applied this test to the alcohol dehydrogenase gene (*Adh*) of *D. melanogaster*, *D. simulans*, and *D. yakuba*, whose substitution data is summarized in Table 12.3. Such a table is tested using Fisher's exact test [231]. On the basis of this test, the authors rejected the null hypothesis that $7/17 \approx 2/42$. In other words, there is a significant excess of fixed differences between the three species. The authors conclude that this is due to adaptive fixation of selectively advantageous mutations [178]. This conclusion agrees with that drawn in Section 12.1 by applying the HKA test to the same locus.

12.5 Minimum Number of Recombination Events

The neutrality tests described so far were all based on the assumption that there has been no recombination within the loci investigated. We now look at a procedure for assessing the validity of this assumption.

Table 12.3. Polymorphism data for the alcohol dehydrogenase gene (*Adh*) of *D. melanogaster*, *D. simulans*, and *D. yakuba*. Data taken from [178].

	Fixed differences	Polymorphisms
Non-synonymous	7	2
Synonymous	17	42

Figure 12.4 shows a pair of chromosomes that differ at two positions along their DNA sequences. Hence, these chromosomes form two distinct haplotypes, `AA` and `TT`. If reciprocal recombination takes place between the two polymorphic loci, two new haplotypes are formed, `AT` and `TA`, leading to a total of four haplotypes. Since under the infinite sites model any position is assumed to mutate only once, the detection of all four possible haplotypes between a pair of single nucleotide polymorphisms is evidence of a recombination event. Counting the number of four-allele pairs of nucleotides should therefore give us an indication of the number of recombination events that have taken place in the evolutionary history of a sample of DNA sequences. However, there are two complications to consider. First, we have no guarantee that representatives of all four haplotypes generated by a recombination event actually end up in our gene sample. The number of four-allele combinations is therefore going to lead to an estimate of a *minimum* number of recombination events, R_{M} [123]. Second, not all pairs of loci that display four alleles are evidence of a recombination event.

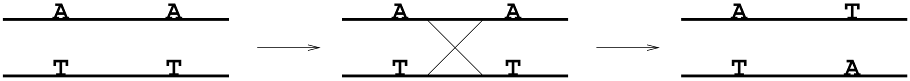


Fig. 12.4. Reciprocal recombination between two polymorphic sites generates the haplotypes `AT` and `TA` out of the original pair of haplotypes `AA` and `TT`.

In order to calculate R_{M} from a set of aligned DNA sequences with m polymorphic sites, we can apply the following algorithm [123]:

1. Construct a matrix $M = (m_{ij})$ that contains for each pair of polymorphic positions i, j an entry of $m_{ij} = 1$ if the pair forms all four possible haplotypes and $m_{ij} = 0$ otherwise.
2. In the upper triangle of M each pair i, j with $m_{ij} = 1$ is interpreted as an open interval (i, j) and a list of such intervals $\{(i_1, j_1), (i_2, j_2), \dots, (i_m, j_m)\}$ is formed, where $i_1 \leq i_2 \leq \dots \leq i_m$.
3. Delete from the list all intervals that are subintervals of another interval.
4. Delete all intervals that overlap with (i_k, j_k), starting at $k = 1$ and continuing to $k = m - 1$.

5. R_{M} is the final number of intervals in the list.

If $R_{\mathrm{M}} > 0$, we know that recombination has taken place. However, we cannot conclude no recombination from $R_{\mathrm{M}} = 0$ because, as noted already, our sample of DNA sequences might not contain all four haplotypes generated by reciprocal recombination.

12.6 Detecting Linkage Disequilibrium

We have seen that the HKA-Test (Section 12.1) is not only based on the assumption of no within-locus recombination, but also that the loci investigated are not linked. There are two meanings of the term “linkage” between two or more loci. First, it might refer to the fact that two loci occur on the same chromosome. Second, it might refer to the fact that two loci are not statistically independent. In the context of hypothesis tests this probabilistic interpretation is usually applied. Our investigation of its implications starts with the simplest case of two bi-allelic loci, $\mathcal{X}$ and $\mathcal{Y}$. The probability of finding allele “1” at locus $\mathcal{X}$ is denoted by p_1 and at locus $\mathcal{Y}$ by q_1. The frequencies of the four possible haplotypes are denoted P_{00}, P_{01}, P_{10}, and P_{11}. Our measure of linkage disequilibrium is then defined as

$$d = P_{00}P_{11} - P_{01}P_{10},$$

which is zero in case of linkage equilibrium. For more than two alleles, all pairs of alleles need to be considered. If more than two loci are investigated, all pairwise comparisons between loci need to be made. This expansion of pairwise comparisons quickly becomes cumbersome, and a summary statistic, the index of association ($I_{\mathrm{A}}^{\mathrm{S}}$), was developed to measure linkage between an arbitrary number of alleles at an arbitrary number of loci [29].

The index of association is based on the number of pairwise differences between a set of haplotypes. The computation of such pairwise differences is illustrated in Figure 12.5. Let V_{D} denote the observed variance of this number and V_{e} its expectation under the null hypothesis of linkage equilibrium. At linkage equilibrium $V_{\mathrm{D}} = V_{\mathrm{e}}$ and, hence, $V_{\mathrm{D}}/V_{\mathrm{e}} - 1$ might be a useful measure of linkage disequilibrium, which is zero at linkage equilibrium [29]. However, $V_{\mathrm{D}}/V_{\mathrm{e}}$ scales linearly with the number of loci analyzed [121]. We therefore define the *standardized* index of association as [108]

$$I_{\mathrm{A}}^{\mathrm{S}} = \frac{1}{L-1}\left(\frac{V_{\mathrm{D}}}{V_{\mathrm{e}}} - 1\right).$$

The value of V_{e} can be estimated from the data as

$$V_{\mathrm{e}} = \sum_{j=1}^{L} h_j \left(1 - h_j\right),$$

where h_j is the probability of drawing two distinct alleles at locus j,

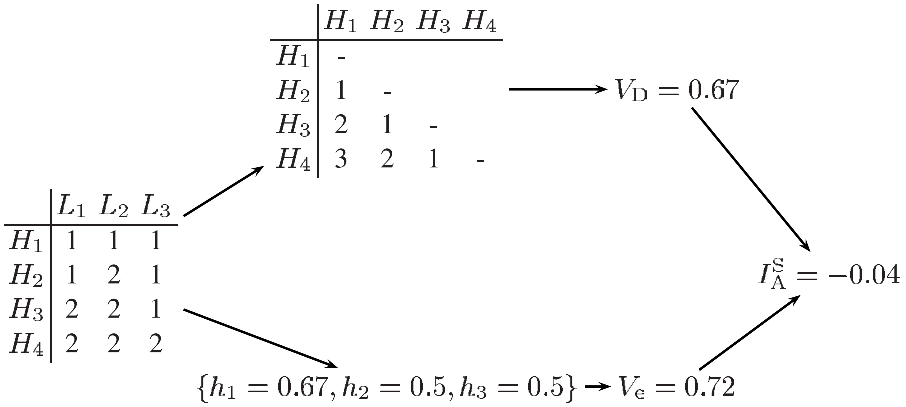


Fig. 12.5. Computation of the index of association ($I_{\mathrm{A}}^{\mathrm{S}}$). Starting from haplotype data for a number of organisms (H) at a number of loci (L), the matrix of pairwise mismatch values is calculated as well as the genetic diversity at each locus (h_i).

$$h_j = \left(\frac{n}{n-1}\right)\left(1-\sum_i p_{ij}^2\right),$$

and where p_{ij} is the frequency in the sample of allele i at locus j.

The significance of a given value of $I_{\mathrm{A}}^{\mathrm{S}}$ can be assessed either by simulation or by assuming that under linkage equilibrium V_{D} is normally distributed. The simulation proceeds by resampling the alleles at each locus without replacement, thereby unlinking the loci [234]. From each resampled data set a new value of V_{D}, V_{D}^i is calculated and the desired significance is the frequency of observing $V_{\mathrm{D}}^i \geq V_{\mathrm{D}}$. This procedure has a run time $O(rn^2)$, where r is the number of resamplings and n the number of haplotypes sampled. The alternative test is based on the assumption that under linkage equilibrium V_{D} is normally distributed. Given this scenario, a formula for the variance of V_{o} needs to be computed once in time $O(L^4)$ [109]. This parametric test is one-sided, like the test based on simulations. The reason for this is that any deviation from the null hypothesis of linkage equilibrium leads to an inflation of V_{D} compared to V_{e}.

The index of association is most popular among microbiologists, because it is easy to obtain the haplotype data on which the test is based from bacteria. In microbiology the application of the index of association has lead to the discovery that bacterial populations have structures ranging from strictly clonal in *Escherichia coli* to panmictic in *Neisseria gonorrhoeae* [175]. However, the test has also been applied to eucaryotes, for example to investigate the population structure of wild barley, *Hordeum spontaneum* [29], or of thale cress, among molecular biologists better known as the model plant *Arabidopsis thaliana* [225].

12.7 Implementations

With the exception of the index of association, the tests discussed in this chapter have been implemented in the software `DnaSP` [211]. This is a freely available program with a user-friendly graphical interface. The index of association has been implemented in the software `LIAN`, which is also freely available [108].

12.8 Summary

Since the inception of the neutral mutation hypothesis in the 1960s, a large number of tests have been devised that assess the validity of various evolutionary scenarios. In this chapter we have covered four tests of neutral evolution, the Hudson-Kreitman-Aguadé (HKA) test, Tajima's D, Fu and Li's test, and the McDonald-Kreitman test. The first three of these are based on arbitrary DNA polymorphism data. Tajima's D as well as Fu and Li's test have the pleasing property that the direction of selection can be read from the sign of the test statistics. In addition, the data for these tests are drawn from a single population, while the HKA test is based on data from two species. Similarly, the McDonald-Kreitman test is based on DNA sequences obtained from two or more species. However, in the case of the McDonald-Kreitman test the input sequences need to be protein coding. All four tests of neutrality described in this chapter assume that there has been no within-locus recombination in the history of the sample. This assumption can be investigated by computing a minimum number of recombination events, R_{M}, based on the four-haplotypes test. If $R_{\mathrm{M}} > 0$ we know that recombination has taken place, while the converse inference cannot be drawn. In addition to the assumption of no within-locus recombination, the HKA test also assumes that the loci investigated are in linkage equilibrium. A convenient measure of linkage disequilibrium between an arbitrary number of loci with two or more alleles is the index of association, $I_{\mathrm{A}}^{\mathrm{S}}$.

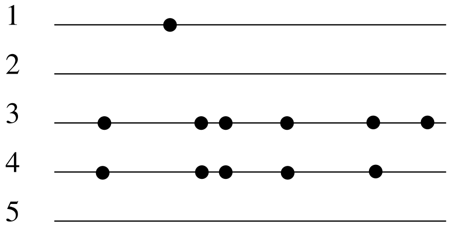
12.9 Exercises and Software Demonstration

12.1. Calculate the harmonic numbers H_4 and $H_4^{(2)}$.

12.2. Draw a genealogy for $n = 4$. What is the number of external and internal branches on that genealogy? What is the general number of external and internal branches in genealogies describing the history of n sequences?

12.3. Use the `bioinformer`-Program `Evolution` → `Coalescent` to familiarize yourself with the relationship between the genealogy of a sample of genes, the distribution of mutations on this genealogy, and the resulting observable polymorphism pattern.

12.4. Here is a stylized haplotype data set:



Draw a possible genealogy that explains this data set.

12.5. Draw an alternative polymorphism pattern that might be generated by the genealogy just uncovered.

12.6. Draw an alternative genealogy that could generate the haplotype pattern shown in Exercise 12.4.

12.7. Given the two haplotypes `AG` and `TC`, what are the haplotypes generated by reciprocal recombination between the two SNPs?

12.8. Consider the following DNA sequence data

```
AGTCGA
AACCCC
TGTTCA
TACTGC
```

and calculate its R_{M}.

12.9. Consider the following multilocus data set

	L_1	L_2	L_3	L_4
H_1	1	1	3	2
H_2	1	1	3	2
H_3	2	2	3	2
H_4	1	4	5	7

and calculate its standardized index of association, $I_{\mathrm{A}}^{\mathrm{S}}$.

A

`bioinformer`

`bioinformer` is an integrated collection of demonstration programs for visualizing a number of the concepts and data structures treated in this book. The following sections give some background to each of the demos, describe the program, and conclude with a tutorial.

A.1 Alignment

A given sequence whose function is unknown can often be annotated by finding a similar sequence of known function among the large number of sequences deposited in public data bases. Searches for sequence similarity are usually based on alignments. The following programs concentrate on pairwise alignment through simple dynamic programming and protein substitution matrices.

A.1.1 Protein Substitution Matrices

Pairs of residues in DNA alignments are traditionally scored by distinguishing only between matches and mismatches. This works reasonably well for DNA sequences but is not applicable to proteins. For a start, the number of mutational steps needed to convert a codon for one amino acid into that for another varies between one and three. In addition, amino acids differ markedly in their physico-chemical properties (Fig. C.4). As a result, amino acid substitutions have highly variable effects on the structure of the protein affected. Since most changes in a protein's structure are deleterious, selection will filter out substitutions with drastic structural effects. Finally, substitution probabilities change over time, making it necessary to construct series of substitution matrices to cover a range of evolutionary time scales. Substitution matrices summarize the $21 \cdot 20/2 = 210$ log-likelihoods of observing a homologous pair of amino acids (cf. Section 2.4).

Description of Program

The program displays the PAM and BLOSUM matrices. On the `PAM tab` PAM matrices are provided for evolutionary distances ranging from 2 PAM to 500 PAM. The user selects the PAM number through a slider which can be manipulated using either the mouse or the ← and → keys. The entries of PAM matrices are expressed in bits and by convention different bit fractions have been used to scale the majority of matrix entries to single digit integers. The user can therefore choose from a range of such scale factors. In addition, the program displays the expected percentage of mismatched residues between proteins separated by an evolutionary distance of x PAM. The implemented calculations are based on the original data [46] and, hence, the matrices may differ in numerical detail from the versions currently available, e.g. as part of the BLAST distribution [10].

On the `BLOSUM tab` three commonly used BLOSUM matrices can be displayed: BLOSUM45, BLOSUM62, and BLOSUM80.

Tutorial

1. Look at the PAM matrix displayed when the program is first launched (use the reset button if necessary) and notice that the values along the main diagonal, which represent match scores, are all positive. This contrasts with the off-diagonal entries, which represent mismatch scores. These are on average negative. Identify the two amino acids with the highest entries on the main diagonal, i.e. the two most conserved amino acids. Can you make biological sense of their high degree of conservation?
2. Slide the `PAM number` to small values. Notice that the expected difference is now equal to the `PAM number`. This is because, say, two mutations per 100 amino acids will on average have caused a 2% difference in the corresponding pairwise comparison.
3. Slide the `PAM number` to very high values. Notice three things:
   a) The `PAM number` now vastly exceeds the `Expected difference`. This is because mutations can hit a given position more than once and this occurs with increasing frequency the more mutations are thrown randomly at a segment of 100 amino acids. At a more basic level, percentages vary between 0 and 100, while the number of mutations per 100 residues (PAMn) can range between 0 and infinity.
   b) The entries in the matrix tend towards 0. The reason for this is that the probability of finding the corresponding pair of residues due to homology becomes in the long run equal to the probability of finding the pair by chance. The ratio of two equal numbers is 1 and $\log 1 = 0$.
   c) A few amino acids retain high entries on the main diagonal even for high PAM values. Identify the two amino acids with the highest entries.
4. Compare the entries between PAM160 and BLOSUM62 for amino acid pairs histidine/alanine, valine/valine, and tryptophan/tryptophan. Do you notice any differences?

A.1.2 Number of Alignments

An alignment can end in three different ways:

```
n n -
- n n
```

where `n` indicates any residue. These three possibilities suggest the following recursion for the number of possible global alignments [253]:

$$f(n,k) = f(n-1,k) + f(n-1,k-1) + f(n,k-1),$$

where $f(n,k)$ is the number of possible alignments between two strings of length n and k. If we further define the boundary condition that there is a single alignment between a sequence of any length and a sequence of length zero, we are in a position to compute the number of possible global alignments (cf. Section 2.5).

The possible number of local alignments is the sum of the number of global alignments for all possible pairs of substrings. A sequence of length n contains $n-m+1$ substrings of length m. Consider as an example two sequences of length two each. Nine pairs of substrings can be formed between these. Four of the substring pairs have length (1,1), yielding three possible global alignments each; another four have length (2,1) or (1,2), yielding five possible global alignments each; one has length (2,2), yielding 13 possible global alignments. The number of possible local alignments is therefore $12 + 20 + 13 = 45$.

Description of Program

The program calculates the number of different global or local alignments that can be constructed for two sequences of lengths between 1 and 400. The user can choose the lengths of the two sequences as well as the alignment mode (global or local).

Tutorial

1. Contemplate the number of global alignments that can be formed between two sequences of lengths 339 and 320.
2. Compare that number to the number of molecules contained in a liter of water: $3.3 \cdot 10^{25}$. How much water is needed to accumulate the same number of water molecules as there are possible alignments between two sequences of lengths 339 and 320?
3. Compute the time it would take to list all possible global alignments between two sequences of length 320 given that the calculation of a single alignment takes, say, 10^{-15} seconds.
4. Compute the time it would take to list all possible local alignments for two sequences of length 320.

A.1.3 Pairwise Alignment

Similar sequences tend to encode similar functions. The search for sequence similarities is therefore the bread and butter of computational biology. This program is concerned with alignment by dynamic programming, which was pioneered for global alignment by Needleman and Wunsch [189]. The local alignment algorithm is due to Smith and Waterman [230].

Description of Program

This program visualizes the dynamic programming matrix used for computing optimal pairwise alignments between two sequences over an arbitrary alphabet. It also visualizes the path through the matrix that corresponds to an optimal alignment. In addition, it displays the alignment, its score, and calculates the number of cooptimal global alignments. Three alignment modes are implemented:

1. global: sequences are assumed to be homologous over their entire length (Section 2.6);
2. overlap: sequences are assumed to have overlapping ends (Section 2.7);
3. local: sequences are assumed to possess only local homology (Section 2.8).

The user can enter any sequence and manipulate the following parameters:

1. match score,
2. mismatch score,
3. gap extension.

Notice that most alignment algorithms implement an affine gap scoring scheme, i.e. the total gap cost, G, is

$$G = g_\mathrm{o} + g_\mathrm{e} \cdot l,$$

where g_o is the gap opening cost, g_e the gap extension cost, and l the length of the gap. In this program the gap opening cost is ignored, i.e. set to zero.

Tutorial

1. Choose the global alignment algorithm.
2. Click on >> to set up the dynamic programming matrix corresponding to the sequences entered.
3. Click again on >> to initialize the dynamic programming matrix.
4. Click on >> to fill in the dynamic programming matrix. Verify that the entry in cell (`A`, `A`) has the expected value.
5. Click one final time on >> to carry out the traceback. The corresponding alignment and its score is shown at the bottom of the panel. In addition, the number of cooptimal alignments is displayed. These are alternative paths through the matrix carrying the same optimal score. There is no algorithmic criterion for distinguishing between these alternatives.
6. Repeat the above steps using the local alignment algorithm.
7. Repeat the above steps using the overlap alignment algorithm.

A.2 Match

Operations based on string matching are an integral part of sequence analysis. The following programs concentrate on exact matching with and without suffix trees.

A.2.1 String Matching

At its most basic, string matching consists of looking for a short pattern in a large text. This is a classical topic of computer science [99]. However, string matching methods are ubiquitous in computational biology, since a core concern of the field is the interpretation of genetic sequence texts [99].

Description of Program

The program implements three methods of string matching:

1. **Naïve string matching** (Section 3.2), which essentially consists of two nested `do` loops, the outer running over the text, the inner over the pattern. Accordingly, the worst case run time of this algorithm is $O(n \cdot m)$, where m and n are the lengths of the pattern and the text, respectively.
2. The **Z-algorithm** (Section 3.3) is a simple string matching algorithm that runs in time linear in the length of the text [99].
3. A **suffix tree** (Section 3.6) is a data structure for indexing a text. Once this data structure is built, it has the astonishing property that a text can be searched in time proportional to the length of the pattern. Building the suffix tree itself only takes time proportional to the length of the text, but the factor of proportionality is rather large [249, 99].

To demonstrate the time requirements of the three algorithms the program displays the time taken for searching and—with the exception of the naïve algorithm, which lacks preprocessing—the time taken for preprocessing. The text window can be freely edited. In addition, three chunks of interesting prose are provided:

1. The sequence of the first genome to be sequenced, that of the bacteriophage ϕX174, which encodes 10 proteins in 5,386 nucleotides [6].
2. The alcohol dehydrogenase (*Adh*) gene of *Drosophila melanogaster*. In 1983 this was the first gene to be subjected to comparative sequencing [152].
3. The text of the paper by Watson and Crick announcing the double helical structure of DNA [256].

Tutorial

Select the genome of ϕX174 and run the pattern search for all three algorithms provided. Observe in particular the variations in preprocessing time.

A.2.2 Suffix Tree

Molecular Biologists often search for sequence patterns in molecular genetic data. Consider for example the text $T =$ `ACCGTC` and the pattern $P =$ `C`. A string search should tell us as efficiently as possible that P occurs in T at positions 2, 3, and 6. Classical results in computer science have lead to algorithms that achieve this in time proportional to the length of T.

Suffix trees are an ideal data structure for improving on this result in cases where the text is stable and needs to be searched repeatedly. Suffix trees are an index structure of the text, which can be built in linear time [249]. Once built, the text can be searched for a pattern in time proportional to the length of the pattern rather than in time proportional to the length of the text (Section 3.6).

Description of Program

The program draws a suffix tree for any input string. On this suffix tree it optionally displays four features:

1. The leaf labels, which are the starting positions of the suffices constructed by concatenating edge labels from the root of the tree to the respective leaf. A suffix of text $T =$ `ACCGTC` starts somewhere inside T and ends at the end of T, for example the suffix `GTC` starts at position 4.
2. The edge labels, which return one of the text's suffices when read from the root to a leaf.
3. The string depth, which indicates the length of the string stretching from the root to the node.
4. The suffix links, which make it possible for a suffix tree to be constructed in linear time [249, 99].

Tutorial

1. Enter a single letter and construct the corresponding suffix tree. Notice that the program always adds the sentinel character `$` to a string before processing it.
2. Add a second letter to the input string that is different to the first letter and construct the suffix tree.
3. Continue adding different letters. Notice that the program is case sensitive.
4. Add a repeated letter and observe the effect on the tree topology.
5. Construct a suffix tree from a string with lots of repetitions. Search it for a pattern by matching from the root. If the pattern is found in the tree, its start positions in the text are given by the leaves in the subtree marked by the end of the match.
6. Notice that the string read from the root to any internal node corresponds to a repeated string in the input sequence. The length of this string is indicated by the `String Depth`. Look for the longest repeat in the input string.

A.2.3 Repeat Searching

Inexact repeats are ubiquitous in complex genomes such as those of mice [40] and men [128]. In addition, exact repeats between two sequences can be used for constructing fast alignment algorithms [48]. As a result, there is a lot of interest in algorithms for detecting repeats in strings (Sections 3.9 and 3.10).

Description of Program

The program uses a suffix tree for locating either repeats of a pre-defined length or for identifying the longest repeat. Sets of repeats are marked by the same background color. The text window can be freely edited. In addition, three chunks of interesting prose are provided:

1. The sequence of the first genome to be sequenced, that of the bacteriophage ϕX174, which encodes 10 proteins in 5,386 nucleotides [6].
2. The alcohol dehydrogenase (*Adh*) region of *Drosophila melanogaster*. In 1983 this was the first gene to be subjected to comparative sequencing [152].
3. The text of the paper by Watson and Crick announcing the double helical structure of DNA [256].

Tutorial

1. Look for the longest repeat in the double helix paper.
2. Look for the next longest repeat in that paper. Is the repeat length correlated with the phrase's importance?

A.2.4 Hash Table

Hash tables are generalized arrays, where instead of using integers as indices, arbitrary objects are utilized. Such indices are usually referred to as *keys* and the object accessed though a key as its *value*. The hashing of query sequences into words of a predefined length is an important step in some fast database search tools, including FASTA [197].

Description of Program

The program takes as input an arbitrary string and computes a table of words of a certain length and their positions in the text. The word length can be varied by the user. The program also displays an indexed version of the input string for easy comparison with the table entries.

Tutorial

1. Hash the default sequence into words of length 1.
2. Hash the sequence into words of increasing length. What is the longest repeat in the input sequence?
3. Enter a short English sentence (e.g. *O, take the sense, sweet, of my innocence!*) and hash it into words of length 3 by either pressing `Enter` or clicking `>`. Do you find a repeated hash key? Compare the result with a hash of `MADAMIMADAM`.

A.2.5 Dotplot

A dotplot is a simple means of comparing two strings (Fig. 4.6). A simple generalization of the kind of dotplot shown in Figure 4.6 is achieved by plotting matches of some length ≥ 1 and this approach has been used to compare sequences on a genomic scale [233]. In addition, dotplots are used to visualize the FASTA search strategy (Section 4.2.2).

Description of Program

The program takes two sequences as input and displays matches of a certain length. The user can enter new input sequences and vary the length of the matches displayed. In addition, the user can randomize the input sequences to observe the length of matches between random sequences.

Tutorial

1. Click > to draw a dotplot of the two alcohol dehydrogenase sequences supplied by the program using a `Match Length` of 11.
2. Observe the matches, indicated as black lines, along the diagonal.
3. Notice the gap in the matches, which is caused by the insertion of the copia transposon in Sequence 1 at that position (cf. header line of Sequence 1).
4. Randomize the two sequences by clicking the die and redraw the dotplot.
5. Vary the `Match Length` to find the longest match between the two randomized sequences.

A.3 Probability

Biological data is often described as "noisy". Its analysis is therefore routinely coupled to probabilistic considerations. The following program demonstrates the application of probability theory in hidden Markov models.

A.3.1 Hidden Markov Model

Hidden Markov models are probabilistic models of sequential data (Section 6.2). Such sequential data could be a stream of phonemes uttered by a speaker or a stretch of nucleotides. In the case of nucleotides we might be interested in inferring the position of genes along the sequence. Say we knew that protein coding regions were G/C-rich compared to non-coding regions. Then we could model our stretch of DNA by postulating two hidden states corresponding to *coding* and *non-coding*. These two states emit nucleotides with different probabilities. Moreover, given that the system is in the coding state, it changes into the non-coding state with some probability and *vice versa*. The set of transition probabilities between the hidden states and emission probabilities attached to each of the hidden states is called the hidden Markov model.

Given such a model and an unannotated stretch of sequence, we can try to guess the locations of coding and non-coding regions. If, on the other hand, we do not know the probability parameters of the model, we can try to infer the model from some "training" data using an arbitrary model as our starting point.

Description of Program

The program allows the user to manipulate the probabilities of a hidden Markov model of a DNA sequence. The model consists of two hidden states and two observable states. The observable states are `A/T` and `G/C`, while the hidden states are indicated solely by different colors. Given this model, the user can generate a DNA sequence with hidden states indicated by color coding. Conversely, the program can guess the most likely sequence of hidden states given simulated DNA sequence data. The inferred hidden states can then be compared to the known states.

In addition, the program can take a simulated DNA sequence and a random hidden Markov model as input and by an iterative procedure infer the most likely model.

Tutorial

1. `Detect Hidden States` Tab
    a) The hidden Markov model on the left consists of a starting point, which changes into one of the two hidden states shown as colored panels with the probabilities indicated on the transition arrows. The hidden states each contain a histogram with two bars indicating the probabilities with which they emit either an `A/T` or `G/C`. When emitting `A/T`, the model emits with equal probability an `A` or a `T`, and similarly for `G/C`. Once the model is in a certain hidden state, it either stays in that state with the probability noted next to the self-referential arrow or changes to the other state with the probability noted next to the arrow pointing to the alternative hidden state. All ten probabilities displayed on the model can be manipulated with the five sliders on the right. Try out all five sliders and observe the effect.
    b) Reset the model by clicking the reset button.

   c) Generate a DNA sequence by clicking `>`. This sequence appears in the lower panel on the lefthand side with the hidden states marked by the same colors as they appear in the model.
   d) Estimate the sequence of hidden states from the data by clicking `?`. The DNA sequence with the estimated annotations appears in the panel on the bottom righthand side. Compare the known hidden states with those estimated from the data.
2. `Estimate HMM` Tab
   a) There are three hidden Markov models displayed here; from left to right they indicate:
      - The model generated in the `Detect Hidden States` tab.
      - A random initial model.
      - The model yet to be estimated. This model is initialized to the same values as the random model on its left.
   b) Generate a new DNA sequence according to the model displayed on the left by pressing `>`.
   c) Estimate a new model by pressing `?`. This may take some time.
   d) Compare the estimated model to the random initial and the true models. Did you get a good fit?
   e) Generate a new random initial model by clicking the die and repeat the model estimation. You will find that model estimation depends to some extent on the initial model. The reason for this is that the estimation algorithm, also known as Baum-Welch algorithm, can easily converge on a local optimum rather than the global optimum, which should be very similar to the true model.

A.4 Evolution

The pattern of polymorphisms detected when comparing homologous sequences is the product of evolution. The following programs demonstrate the construction of phylogeny as well as the forward and reverse simulation of genes in populations.

A.4.1 Phylogeny

The human interest in reconstructing the ancestry of powerful families is at least as old as the oldest literary documents. For example, *Genesis* 5 lists the descendants of Adam down to Noah and *Genesis* 10 recounts the descendants of Noah's three sons, who go on to populate the earth. The preceding description of creation (*Genesis* 1) is remarkably free of such genealogical thinking. However, at its most basic level a phylogeny is simply the generalization of a family tree (Chapter 8).

Description of Program

The program takes as input a set of aligned sequences in FASTA format and seven primate mitochondrial DNA sequences are provided as default input. The program computes the pairwise distances between these sequences. Users can choose between three different distance measures:

1. number of mismatches;
2. normalized number of mismatches, that is, the number of mismatches divided by the length of the alignment;
3. Jukes-Cantor distance [130].

A simple phylogenetic tree is reconstructed from these distances using the UPGMA (unweighted pair group method with arithmetic means) method (Section 8.4.1). In addition, the program demonstrates the application of the bootstrap in phylogeny reconstruction (Section 8.8).

Tutorial

1. `Distance Matrix` Tab
   a) Compute the pairwise number of `Mismatches` between the displayed sequences by pressing >. Notice that the first number in the panel containing the distance matrix is the number of taxa analyzed. This corresponds to the file format used in PHYHLIP, a widely used package for calculating phylogenies [72].
   b) Count the number of mismatches between the human and chimpanzee sequences and compare your result with the corresponding entry in the distance matrix.
   c) Compute the `Normalized Mismatches` and make a note of the distance between human and gibbon.
   d) Compute the `Jukes-Cantor` distance. Compare the Jukes-Cantor distance between human and gibbon with the normalized number of mismatches for this pair of taxa.
2. `Phylogeny` Tab
   a) Compute the phylogeny from the distance data (>>).
   b) Display the branch lengths on the tree.
   c) Compare the branch lengths with the distances in the distance matrix.
3. `Bootstrap` Tab
   a) Generate a bootstrap sample from the original data. This is done by sampling with replacement columns from the original alignment.
   b) Compute the distance matrix for the bootstrap data (>>).
   c) Generate the phylogenetic tree for the new distance matrix (>>). Is the tree topology the same as for the original data set?
   d) Repeat the process of bootstrapping and observe the range of tree topologies generated (>>).
   e) Click the gearwheel to animate the generation of bootstrapped trees. Notice that you can vary the animation speed using the `Step Time` slider.

A.4.2 Drift

Random changes of allele frequencies during evolution are called drift. This contrasts with changes in allele frequencies due to a host of other possible factors, the most interesting of which is selection. Since drift is easy to model, it often serves as a point of departure for an investigation of the evolutionary forces acting on a population (Section 10.3).

Description of Program

The program consists of two modules, *Drift without Mutation* and *Drift with Mutation*.

1. Drift without Mutation: This demonstrates evolution in a two-allele system with alleles A and a. The user can choose a population size and an initial frequency of the A-allele. By pressing the *Run*-button (>) the simulation starts showing the frequency of the A-allele and the genetic diversity, H, as a function of time measured in generations. Repeated pressing of the *Run*-button launches new runs of the simulation and a concomitant entry in the legend that displays the population size.
   All simulations can be stopped by pressing the *Stop*-button and the panel is reset by pressing the *Reset*-button. If a simulation runs off the graph, you can still follow it by dragging the graph panel with the left mouse button to the left. You can also zoom into the graph by selecting a region of interest; to zoom out again the same action of dragging the graph panel with the left mouse button to the left is used. Furthermore, by right-clicking the graph a number of layout options are opened.
2. Drift with Mutation: Here the user can set a population size and a mutation rate ranging from 0 to 0.01. Pressing the *Run*-button starts the simulation under the infinite alleles model. The genetic diversity of the population is shown as a function of time in generations. In addition, the expected genetic diversity is also shown. Each click on the *Run*-button starts a new simulation, while pressing the *Stop*-button stops all of them. The *Reset*-button returns the panel to its initial state.

Tutorial

1. `Drift without Mutation` Tab
   a) Using the default parameters, start a simulation run (>). Notice that the A-allele quickly goes either to fixation ($\text{Frequency}(A) = 1$) or extinction ($\text{Frequency}(A) = 0$) and that both events correspond to a genetic diversity of zero.
   b) Set the initial frequency of A to 0.1 and observe the frequency of runs that go to fixation. Compare this with $\text{Frequency}(A) = 0.5$.

   c) Reset the program, increase the population size to 100, set $\mathrm{Frequency}(A) = 0.5$, and start a new simulation run. This time it takes longer on average for A to either reach fixation or go extinct.
   d) Leave the population size at 100 but vary $\mathrm{Frequency}(A)$ and observe the frequency of fixation in a number of runs.
2. `Drift with Mutation` Tab
   a) Using the default parameters, start one simulation run (>). Recall that without mutation the frequency of a given allele eventually either reaches unity (fixation) or zero (extinction). Both values correspond to a genetic diversity of zero at which the locus in the population gets stuck in the long term. In contrast, in the presence of mutation, genetic diversity can always recover from a crash to zero as long as the mutation rate, μ, is greater than zero.
   b) Continue to observe the first simulation run and notice the horizontal line indicating the expected genetic diversity. This is computed as

$$\bar{H} = \frac{2N\mu}{2N\mu + 1},$$

   where N is the population size. Here N is multiplied by 2 rather than by 4, because the program simulates a haploid population, where the population size is equal to the number of genes.
   c) Stop the present simulation and reset the program. Change the population size and mutation rate to, say, $N = 200$ and $\mu = 10^{-3}$, and start another simulation run. Notice two things: (i) the observed genetic diversity fluctuates strongly around its expectation and (ii) the values of H when going from one generation to the next tend to be similar to each other rather than being drawn randomly from the full distribution of possible values. The values of H are therefore said to be *autocorrelated*.

A.4.3 Wright-Fisher

The Wright-Fisher model of neutral evolution consists of a population of constant size N that evolves by resampling with replacement. In its original formulation this model moved forward in time (Section 10.3). However, it can also be analyzed backward in time and this perspective forms the basis of coalescent theory (Section 11.3). The aim of this module is to visualize descendants and ancestors under the Wright-Fisher model using the original one-parent genealogy as well as the more familiar, though genetically largely irrelevant, two-parent genealogy.

Description of Program

The program simulates the evolution of a Wright-Fisher population. It displays the evolving entities and the lines of descent that connect them between the generations. We can think of the evolving entities as homologous genes in the one-parent case or as individuals in the two-parent case. The user can choose between these scenarios, as well as select the number of generations simulated and the size of the population.

1. One-Parent Genealogy: By default the program displays a "tangled" version of the lines of descent, which results from letting genes in generation g randomly pick their parent (one per gene) in generation $g + 1$. A disentangled version of this graph is generated by checking `Disentangle`. The offspring of individual genes can be visualized forward in time by clicking genes in generation 1. Similarly, the ancestry of individual genes is displayed by clicking them in the present generation. In coalescent theory the most recent common ancestor of a sample of genes is the point at which the simulation stops. In our program the most recent common ancestor of the selected genes is marked in red. Note that it is possible that a sample of genes has not yet reached its most recent common ancestor.
2. Two-Parent Genealogy: In the two-parent mode only color-coded invividuals are displayed initially. Black individuals are those that have left some descendants in the present generation; red individuals are universal ancestors and blue individuals have left no descendants in the present generation. Like in the one-parent mode, clicking on individuals in the present or in the first generation displays an individual's ancestors and descendants, respectively. Notice that individuals pick their parents randomly in the preceding generation. As a consequence, there is a $1/N$ probability that an individual picks the same parent twice. This corresponds to the original model and can be reconciled with common sense if we reinterpret a dot as a couple rather than an individual [37]. Most couples have two couples as parents, but occasionally the children of one couple mate among each other.

Tutorial

One-Parent Genealogy

1. Using the default parameters, run the program in the one-parent mode once. Red numbers indicate generations, black numbers label individuals. Follow a few of the descendants of individual genes by clicking on them in generation 1. Notice that in quite a few cases genes leave no descendants in the present.
2. Follow the ancestry of individual genes by clicking on them in the present generation. Observe that the position of the most recent common ancestor shifts depending on the genes selected.
3. Disentangle the lineages. This makes the graph easier to read. Notice also that the order of the individuals in the last generation has changed due to the disentanglement. Again follow the descendants and the ancestors of individual genes.
4. Run a few simulations. Notice that in the disentangled mode it takes on average $2N$ generations for the rightmost and the leftmost gene in the present generation to find their most recent common ancestor. In contrast, all remaining genes in the population coalesce on average twice as fast.

Two-Parent Genealogy

1. Run the program once in the two-parent mode using the default parameters. Notice that the layer of generations containing black dots, i.e. individuals with some

descendants in the present, is rather thin. Red dots, that is universal ancestors, appear early on and start to dominate the population further back in time.

2. Click on an individual in generation 1 to visualize its descendants. Notice that blue individuals may or may not have their lineage terminated in generation 1.
3. Click on individuals in the present generation to visualize their ancestors. The ancestors of different individuals start to rapidly overlap.
4. The one-parent and two-parent modes of the program are based on the same underlying population. Hence by switching between the two modes, you can observe that universal ancestors need not have contributed genetic material to a given descendant.

A.4.4 Coalescent

When going backward in time, gene lineages *coalesce* at the point where they diverged from their last common ancestor. Hence the tree structure tracing back the genealogy to the most recent common ancestor of a sample of genes is called the *coalescent* and the associated theory is referred to as *coalescent theory* (Section 11.3).

Description of Program

The program's toolbar allows the manipulation of the sample size and of the population parameter theta (θ). This is the scaled mutation rate

$$\theta = 2N\mu,$$

where N is the size of a haploid population and μ the mutation rate. This parameter has a simple relationship to the expected number of segregating sites, S, found in a sample of DNA sequences [258]:

$$S = \theta \sum_{i=1}^{n-1} \frac{1}{i}, \tag{A.1}$$

where n is the sample size. Notice that for a sample of two sequences the number of segregating sites, also known as single nucleotide polymorphisms, SNPs, is equal to θ. The program demonstrates the creation of sequence samples with mutations.

Tutorial

1. Click on >> to generate a genealogy. Notice the time scale measured in units of $2N$ generations.
2. Click on >> to throw mutations onto the genealogy. The mutations are displayed as red squares and their average number is given by Equation (A.1).

3. Click on `>>` to generate the haplotypes, or sequences, that correspond to the displayed genealogy with mutations. The sequences are stylized as labeled black lines with the positions of the mutations indicated as before by red squares. Each mutation on the genealogy corresponds to a segregating site, i.e. a polymorphic column of haplotype positions. Notice that mutations affect all the sequences that are located below it in the genealogy. Conversely, mutations on the outer branches of the genealogy only appear in a single sequence. These are also known as singletons and on average they make up

$$1/\sum_{i=1}^{n-1} 1/i$$

   of all the mutations [192].
4. Click the gearwheel to animate the display of random genealogies. Notice that you can regulate the speed of animation using the `Step Time` slider.

B

Probability

Statistical reasoning as well as simple probability calculations crop up everywhere in computational biology. In the following we give a very brief summary of these topics. A more detailed account can be found in the textbook on statistical bioinformatics by Ewens and Grant [66]. For an introduction to the subject of probability theory the reader is referred to the classical textbook by Feller [68].

Random Events and Their Probabilities

Many biological processes have a random element, that is, their outcome is uncertain. But even processes with uncertain outcome can be described quantitatively if we observe enough of them under constant conditions. For example, it is uncertain whether a boy or a girl will be born, but if we observe enough births we find that 51.5% of newborns are boys.

An *experiment* is a repeatable process with uncertain outcome. The outcome of an experiment is called an *elementary event*. As an example consider the experiment "dice rolling" and let E be the set of all possible and mutually exclusive elementary events of this experiment, that is, $E = \{1, 2, .., 6\}$.

Every subset $A \subseteq E$ is called an *event*. An event A occurs if and only if one of the elementary events constituting A occurs. In the above example $A = \{3, 4, 5, 6\}$ corresponds to the event of obtaining a number greater than 2 when rolling a die. Every event A has a probability $P(A)$ attached to it, $0 \leq P(A) \leq 1$.

Addition Rule

The probabilities of mutually exclusive events are additive. To put this more formally, let $A_1, A_2, \ldots, A_n$ be random events, $A_i \cap A_j = \emptyset$ for $i \neq j$, then

$$P\left(A_1 \cup A_2 \cup \ldots \cup A_n\right) = P\left(A_1\right) + P\left(A_2\right) + \ldots + P\left(A_n\right), \tag{B.1}$$

which is also known as the addition rule. As an example consider the probability of obtaining a number greater than 2 when rolling a fair die, i.e. one where each elementary event A_i has the same probability, $P(A_i) = 1/6, i = 1, ..., 6$. According to the addition rule, the probability in question is then $P(A_3 \cup A_4 \cup A_5 \cup A_6) = 2/3$.

Conditional Probabilities

The conditional probability $P(A|B)$ is the probability of event A given that event B has already occurred. This is defined as the probability of both A and B occurring divided by the probability of B:

$$P(A|B) = \frac{P(A \cap B)}{P(B)};\ P(B) \neq 0. \tag{B.2}$$

Independent events A, B have the property that $P(A|B) = P(A)$, which together with Equation (B.2) leads to the multiplication rule

$$P(A \cap B) = P(A)\,P(B)\,. \tag{B.3}$$

For example, the probability of obtaining an even number and a number greater than 2 when rolling two dice is $1/2 \cdot 2/3$.

Bayes' Formula

Let the certain event E consist of n pairwise independent events, that is, $E = A_1 \cup A_2 \cup \ldots \cup A_n$ and $A_i \cap A_j = \emptyset$ for $i \neq j$; then we can represent any random event B as $B = (A_1 \cap B) \cup (A_2 \cap B) \cup \ldots \cup (A_n \cap B)$. According to the addition and multiplication rules expressed in Equations (B.1) and (B.3), we obtain from this

$$P(B) = \sum_{i=1}^{n} P(A_i)\,P(B|A_i)\,.$$

The latter expression is also called the *total probability* of event B. Instead of inferring the probability of B we can derive the probability of A_i given B, which is also known as the *posterior* probability of A_i. This leads to Bayes' formula:

$$P(A_i|B) = \frac{P(A_i)\,P(B|A_i)}{\sum_{j=1}^{n} P(A_j)\,P(B|A_j)}. \tag{B.4}$$

As an example of the application of Bayes' formula consider playing in an occasionally dishonest casino where 99% of the dice are fair, but the remaining 1% return a 6 in 50% of cases. Imagine you roll a die three times and obtain a six each time. Are you dealing with one of the biased dice?

$$P(\text{"biased die"}|\text{"3 rolls of 6"}) = \frac{1/100 \cdot (1/2)^3}{1/100 \cdot (1/2)^3 + 99/100 \cdot (1/6)^3} = 0.21.$$

Probably not.

Random Variables and Their Distributions

A *(discrete) random variable* is a numerical quantity, which, as a result of a random experiment, can take any of some discrete set of possible values. For example, the number of CG dinucleotides observed in 1 kb of nucleotide sequence is a variable that fluctuates randomly between different 1 kb fragments. The possible values in this example range between 0 and 999. Associated with a random variable is its distribution. It assigns probabilities to the possible (subsets of) values of a random variable. Random variables are customarily abbreviated by capital letters $X, Y, \ldots$, whereas particular values are usually abbreviated by small letters $x, y, \ldots$. Thus, the expression $\text{Prob}(X = x) = p$ means that the "probability that the random variable X takes value x is equal to p". Important characteristics of a distribution are its *moments*. The k-th moment of random variable X is defined as $E(X^k) = \sum_x x^k \, \text{Prob}(X = x)$, where the summation ranges over all possible values of X. The first moment is also known as the *mean* or expectation. The *variance* is the difference between the second moment and the squared first moment. For instance, for random variable X with mean $E(X)$, the variance is $V(X) = E(X^2) - E^2(X)$.

In genetics, a frequently encountered distribution is the *Poisson* distribution. A discrete random variable X which can take integer values $0, 1, 2, \ldots$ is said to be Poisson distributed with parameter λ, if

$$\text{Prob}(X = x) = \exp(-\lambda)\frac{\lambda^x}{x!},$$

where $\lambda > 0$ is a real number. The mean and variance of a Poisson distributed random variable are $E(X) = V(X) = \lambda$.

In Chapter 10 we use the concept of conditional expectations. The *conditional expectation* of a random variable X given that some other random variable takes some value $Y = y$ is written as $E(X|Y = y)$. It is the first moment of X under the conditional probability distribution $\text{Prob}(X = x|Y = y)$.

A more complete list of the basic properties of random variables, including continuous random variables, and conditional expectations can be found in the textbook by Karlin and Taylor [136].

Parameters and Their Estimators

It is important to distinguish between parameters and their estimators. Parameters are the usually unknown properties of populations. In contrast, estimators are defined with respect to a sample drawn from the relevant population in order to estimate a parameter. For example, the variance of a sample, s^2, is a possible estimator of the population variance σ^2.

Exercises

B.1. Consider a disease-causing recessive genotype that occurs in one individual out of 10,000. What is the expected frequency of the corresponding disease phenotype?

B.2. Imagine a genetic defect which increases the risk of colon cancer in old age and which affects one in a million individuals. A test for this mutation has become available recently. This test has a 0.01% false positive rate. Based on Bayes' formula, you decide not to undergo the test. Why not?

B.3. The genome of the mustard weed *Arabidopsis thaliana* contains approximately $120 \cdot 10^6$ nucleotides. Let $\mu = 10^{-8}$ be the rate of mutation per nucleotide per generation. How many mutations do you expect to occur per generation throughout the genome?

B.4. Assuming that the number of mutations is Poisson distributed with a rate of the expected number of mutations just calculated, what is the probability that the genome of *A. thaliana* has accumulated at least one mutation after one generation?

C

Molecular Biology Figures and Tables

This book is centered on the analysis of nucleic acid and protein sequences.We therefore survey in the following a few fundamental properties of the constituents of these sequences, nucleotides and amino acids. For more details on the current state of molecular biology we refer the reader to Watson's well-known textbook [255], while Judson has written a vivid history of the early days of molecular biology [129]. This history is brought up to date in Watson's popular survey of all that is DNA [254].

Nucleic Acids

DNA and RNA are both nucleic acids and Table C.1 summarizes the nomenclature of their constituents: *nucleosides* consist of a base and a pentose sugar moiety, while *nucleotides* consist of a nucleoside plus a phosphate group. The energy-carrying molecule adenosine triphosphate (ATP) is an example of a nucleotide. However, in computational biology "nucleotide bases" is often used synonymously with "nucleotides". The carbon atoms of the central sugar molecule in nucleotides are numbered as shown in Figure C.1. The phosphate groups link the 3' and the 5' carbons of the sugars into long polymers. One corresponding monomer of DNA and RNA is shown in Figure C.2. The main difference between RNA and DNA is that in RNA the sugar moiety has an extra hydroxyl group at the 2' position. This makes RNA much more reactive than DNA. In addition, uracil replaces in RNA the chemically similar thymine found in DNA. In the cell both molecules can only be synthesized in the direction 5'→3'. The usual form of RNA is single stranded, while DNA consists normally of two antiparallel complementary strands that assemble by specific hydrogen bonding between `A`/`T` and `G`/`C` base pairs into the familiar double helix (Figure C.3).

When sequencing DNA molecules, it is sometimes not possible to precisely determine the identity of a given base. The set of ambiguity codes shown in Table C.2 is used to describe all possible base identifications.

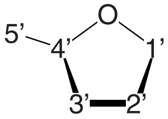


Fig. C.1. Numbering of carbon atoms in the ribose moiety of nucleotides.

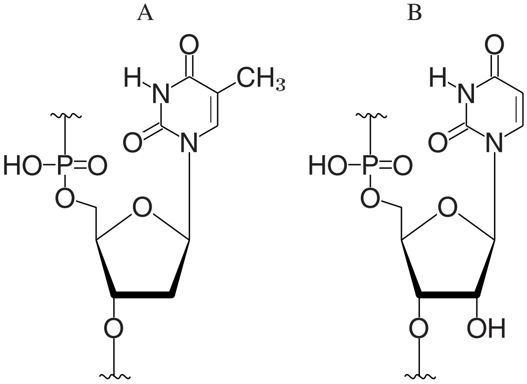

A
B
O
H
N
CH3
O
N
HO–P=O
O
O
O
OH

Fig. C.2. Example of a monomer of DNA (**A**) and RNA (**B**) molecules. The bases can vary from position to position.

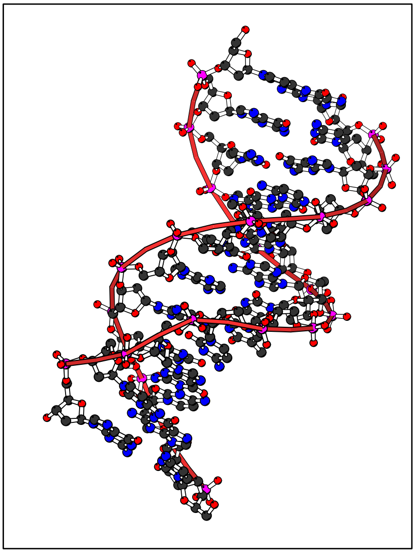


Fig. C.3. Structure of DNA.

Table C.1. Nomenclature of nucleic acid constituents. The cut-off bond (∼) is attached to the ribose moiety.

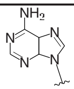
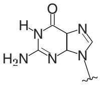
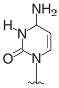
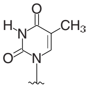
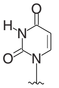
Base	Structure	Chemical Class	Nucleoside	Symbol
adenine		purine	adenosine	`A`
guanine		purine	guanosine	`G`
cytosine		pyrimidine	cytidine	`C`
thymine		pyrimidine	thymidine	`T`
uracil		pyrimidine	uridine	`U`

Table C.2. Ambiguity codes for nucleotides as defined by the International Union of Biochemistry and Molecular Biology (IUBMB).

Symbol	Meaning
A	A
B	TGC
C	C
D	TGA
G	G
H	TCA
K	TG
M	CA
N	TGCA

Symbol	Meaning
R	GA
S	GC
T	T
U	T
V	GCA
W	TA
X	TGCA
Y	TC

Proteins

Proteins consist of amino acids and Table C.3 lists the symbols of the 20 amino acids specified by the genetic code. In Figure C.4 these amino acids are classified according to their physico-chemical properties.

Table C.3. The one letter and three letter codes for amino acids.

One	Three	Name	One	Three	Name
A	Ala	alanine	M	Met	methionine
C	Cys	cysteine	N	Asn	asparagine
D	Asp	aspartic acid	P	Pro	proline
E	Glu	glutamic acid	Q	Gln	glutamine
F	Phe	phenylalanine	R	Arg	arginine
G	Gly	glycine	S	Ser	serine
H	His	histidine	T	Thr	threonine
I	Ile	isoleucine	V	Val	valine
K	Lys	lysine	W	Trp	tryptophan
L	Leu	leucine	Y	Tyr	tyrosine

Proteins are synthesized by a fusion of the amino and the carboxyl groups of the amino acids concerned. This is depicted in Figure C.5 and leads to the creation of an N-terminus and a C-terminus in the resultant protein. Cellular protein synthesis proceeds in the direction N→C as the ribosome moves along the messenger RNA in the direction 5'→3'.

Genetic Code

The genetic code links the information contained in nucleotide sequences to the primary structure of proteins. Table C.4 summarizes the “standard” genetic code. Note that small variations on this standard code exist in the mitochondrial codes of a wide variety of organisms, including members of the metazoans (e.g. vertebrates), fungi (e.g. *Saccharomyces* spp.), and green plants, including land plants. In addition, variant nuclear codes have been described for members of the fungi (e.g. many *Candida* spp.), green algae (e.g. *Acetabularia* spp.), ciliates, diplomonads (e.g. *Giardia* spp.), and firmicutes (e.g. *Mycoplasma spp.*) [150]. These variant codes can be looked up at `www.ncbi.nlm.nih.gov/Taxonomy/Utils/wprintgc.cgi`.

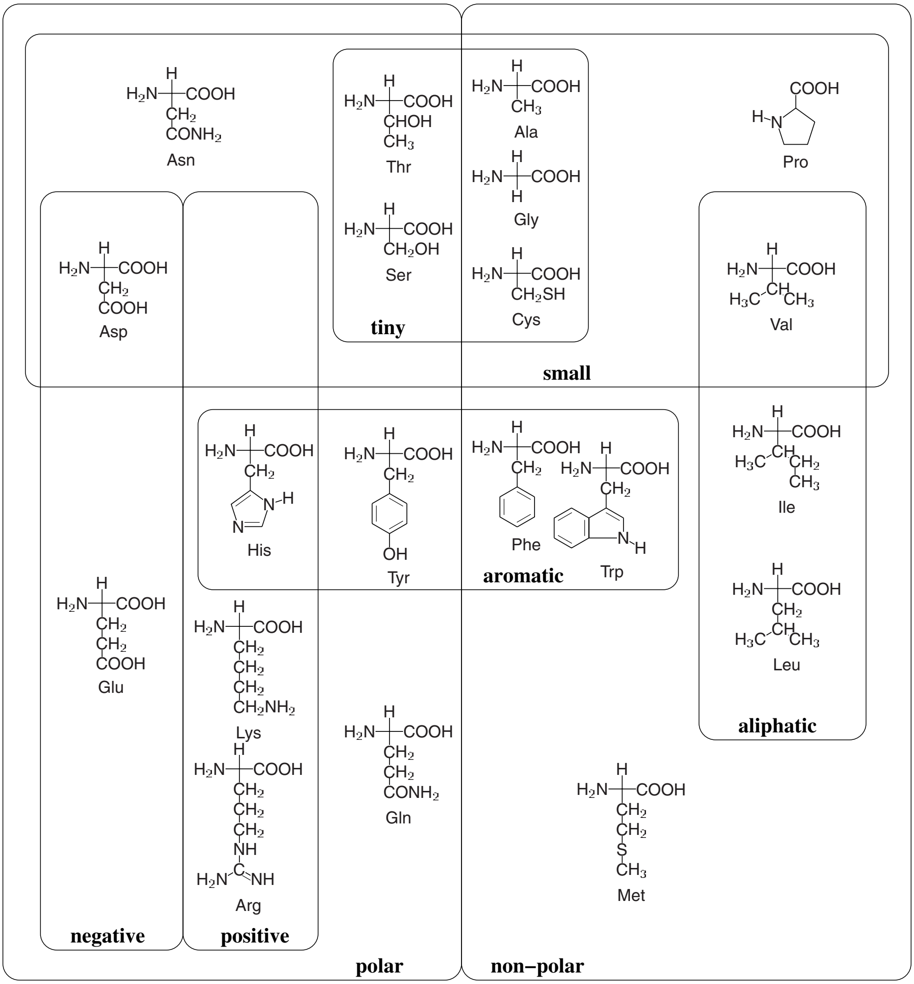

Asn
Thr
Ser
Ala
Gly
Cys
Pro
Asp
Val
tiny
small
His
Tyr
Phe
Trp
aromatic
Ile
Leu
aliphatic
Glu
Lys
Arg
Gln
Met
negative
positive
polar
non–polar

Fig. C.4. Classification of the 20 proteinogenic amino acids according to their physicochemical properties.

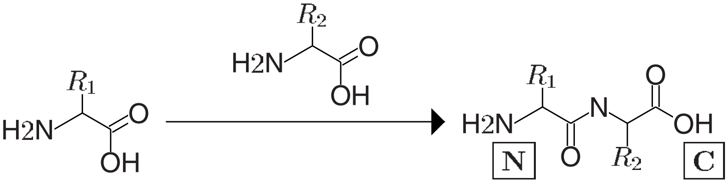

R_1
H2N
O
OH
R_2
N
R_1
R_2
N
C

Fig. C.5. Synthesis of a dipeptide from two amino acids. R_1 and R_2 are side chains drawn from the repertoire depicted in Figure C.4. $\boxed{\text{N}}$ and $\boxed{\text{C}}$ refer to the N- and C-terminus of the peptide, respectively.

Table C.4. The standard genetic code. If your have trouble remembering the order of the bases, then “Think Carefully About Genes”.

5’ end	second position				3’ end
	T	**C**	**A**	**G**	
T	Phe/F	Ser/S	Tyr/T	Cys/C	**T**
					C
	Leu/L		Ter/*	Ter/*	**A**
				Trp/W	**G**
C		Pro/P	His/H	Arg/R	**T**
					C
			Gln/Q		**A**
					G
A	Ile/I	Thr/T	Asn/N	Ser/S	**T**
					C
			Lys/K	Arg/R	**A**
	Met/M				**G**
G	Val/V	Ala/A	Asp/D	Gly/G	**T**
					C
			Glu/E		**A**
					G

D

Resources

In this book we have used and/or referred to a number of key bioinformatics software packages and databases. The ones we use most frequently are listed in Tables D.1 (software) and D.2 (databases). However, our lists reflect only a small portion of what is regularly used by working computational biologists. A good starting point to explore the rapidly changing world of bioinformatics software and databases are the software and database directories maintained by the journal *Nucleic Acids Research* on their website `nar.oupjournals.org`.

Table D.1. A selection of bioinformatics applications.

Program	Purpose	Reference	Website
BLAST	Database search	[9]	`www.ncbi.nlm.nih.gov/BLAST`
`clustalw`	Multiple sequence alignment	[246]	`ftp-igbmc.u-strasbg.fr/pub`
EMBOSS	Various bioinformatics tasks	[209]	`emboss.sourceforge.net`
`DnaSP`	Analyze DNA sequence polymorphisms	[211]	`www.ub.es/dnasp`
FASTA	Database search	[197]	`fasta.bioch.virginia.edu`
`GenScan`	Gene prediction	[30]	`genes.mit.edu/GENSCAN.html`
`gff2aplot`	Plotting sequence comparisons	[2]	`genome.imim.es/software`
`ms`	Coalescent simulations	[122]	`home.uchicago.edu/~rhudson1`
`MUMmer`	Genome alignment	[48]	`www.tigr.org/software`
PHYLIP	Phylogeny reconstruction	[72]	`evolution.genetics.washington.edu`
`SGP-2`	Gene prediction	[196]	`www1.imim.es/software/sgp2`
`Slam`	Gene prediction	[193]	`bio.math.berkeley.edu/slam`

Table D.2. A selection of bioinformatics databases.

Name	Purpose	Website
GenBank	primary sequence repository	`www.ncbi.nlm.nih.gov`
EMBL	primary sequence repository	`www.ebi.ac.uk/embl`
DDBJ	primary sequence repository	`www.ddbj.nig.ac.jp`
SwissProt	curated protein sequences	`www.expasy.org/sprot`
PDB	macromolecular structures	`www.rcsb.org/pdb`
INTERPRO	protein motifs	`www.ebi.ac.uk/interpro`

Answers to Exercises

Chapter 2

2.1 $5 \cdot 1 + 2 \cdot -3 + -5 + 3 \cdot -2 = -12$

2.2

$$1 - 4 \cdot \left(\frac{1}{4}\right)^2 = 3/4$$

2.3 $1 - (0.346^2 + 0.158^2 + 0.159^2 + 0.337^2) \approx 0.72$

2.4 $4 + 0 + 11 + 6 + 9 + 9 - 1 = 38$

2.6

1. First, we compute the matrix $R_{ij} = M_{ij}/p_i$, that is, $R_{ii} = 0.97/0.25 = 3.88$ and $R_{i \neq j} = 0.04$. Then we compute the rounded log-odds scores, $\iota_{ij} = \text{round}\,(\log_2 R_{ij})$ and get $\iota_{ii} = 2$ and $\iota_{i \neq j} = -5$.
2. We start by computing $M' = M \cdot M$. For our example, $M'_{ii} = 0.97^2 + 3 \cdot 0.01^2 = 0.9412$ and $M'_{i \neq j} = 2 \cdot 0.97 \cdot 0.01 + 2 \cdot 0.01^2 = 0.0196$. From this we compute the rounded log-odds scores of $\iota'_{ii} = 2$ and $\iota'_{i \neq j} = -4$.

2.7 In this case clustering does not change the final result, so the answer is shown in Figure 2.11B.

2.8 Expected score:

$$\frac{4 \cdot 3 + 12 \cdot -5}{16} = -3$$

The scoring scheme can be applied in local alignment algorithms because the expected score is < 0.

2.9 129

2.10 $\{\texttt{A}, \texttt{C}, \texttt{G}, \texttt{AC}, \texttt{CG}, \texttt{ACG}\}$

2.11 The number of pairs of substrings is

$$\frac{4 \cdot 3}{2} \cdot \frac{5 \cdot 4}{2} = 60.$$

2.12 Let a_{ij} be the number of global alignments between two sequences of lengths i and j. Then the sought number of alignments is

$$\sum_{i=1}^{3} \sum_{j=1}^{4} \binom{i+1}{2} \binom{j+1}{2} a_{ij} = 808.$$

2.13 The hypothetical shotgun sequencing projects has the following parameters:

1. Coverage:
$$\frac{82500 \cdot 500}{4639675} = 8.89.$$
2. Unsequenced nucleotides: $4639675 \cdot e^{-8.89} \approx 639$.
3. Coverage for one unsequenced bp: $\ln(4639675) = 15.35$.

2.15

1. `ACCGTT--`
   `A--GTTCA`
2. `ACC-GTT--`
   `---AGTTCA`

2.16 Here is an algorithm for calculating the number of cooptimal alignments:

```
Require: $f$ //filled $m \cdot n$ dynamic programming matrix
Require: $c$ //empty $m \cdot n$ dynamic programming matrix
Ensure: number of cooptimal alignments
  for all $1 \leq i \leq m$ do
    for all $1 \leq j \leq n$ do
      $c(i, j) \leftarrow 0$ //initialize $c$ to zero
  $c(m, n) \leftarrow 1$ //initialize start point of traceback
  for $i \leftarrow m$ to $1$ do
    for $j \leftarrow n$ to $1$ do
      if $f(i, j)$ points back to $f(i-1, j)$ then
        $c(i-1, j) \leftarrow c(i-1, j) + c(i, j)$
      if $f(i, j)$ points back to $f(i-1, j-1)$ then
        $c(i-1, j-1) \leftarrow c(i-1, j-1) + c(i, j)$
      if $f(i, j)$ points back to $f(i, j-1)$ then
        $c(i, j-1) \leftarrow c(i, j-1) + c(i, j)$
  return $c(1, 1)$
```

2.18 Introduce a new variable, c, for *cutoff*. Let $c = 0$ for local alignments and $c = -\infty$ for global alignments. Then we can write the global/local recursions:

$$
\begin{aligned}
F(i,0) &= \max(i \cdot g_{\mathrm{e}}, c)\\
F(0,j) &= \max(j \cdot g_{\mathrm{e}}, c)\\
F(i,j) &= \max(F(i,j-1) + g_{\mathrm{e}}, F(i-1,j-1)\\
&\qquad + s(S_1[i], S_2[j]), F(i-1,j) + g_{\mathrm{e}}, c).
\end{aligned}
$$

Chapter 3

3.2 Start with a standard dynamic programming matrix for comparing two sequences. The first row and first column are initialized to zero, matches scored as 1, mismatches as $-\infty$. Then

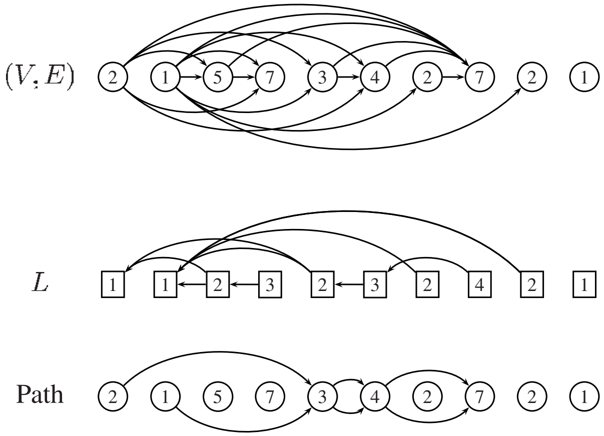
$$F(i,j) = \max(F(i-1,j-1) + s(S_1[i], S_2[j]), 0).$$

Now we simply have to find the cell with the highest entry, which gives us the length of the longest exact match between the two sequences. Notice that no traceback is required for uncovering the match, its position as well as its length suffice for looking it up in the original sequence. This algorithm has a run time $O(|S_1| \cdot |S_2|)$. Application of a generalized suffix tree reduces this to $O(|S_1| + |S_2|)$.

Chapter 4

4.1 Sensitivity = 35/40; specificity = 28/35.

4.2 Here is a summary of the solution strategy:



Start by converting the given sequence of numbers into a graph mapping all possible transitions. As shown in the top row, this graph consists of a set of n vertexes, V, and a set of edges, E, $\forall(i,j) \in E : i < j$. Now we fill in the array $L(i), 1 \leq i \leq n$, where $L(i)$ is the length of the longest subsequence ending at vertex $V(i)$, using the dynamic programming algorithm

```
for $j \leftarrow 1$ to $n$ do
    $L(j) \leftarrow 1 + \max(L(i) : (i,j) \in E)$
```

In order to actually obtain the set of vertexes corresponding to $\max(L(i))$, we store for each $L(j)$ one or more back pointers to $\max(L(i) : (i,j) \in E)$ (cf. middle row). By starting from the maximal entry in L and working backwards we obtain the two cooptimal paths consisting of four nodes each shown in the bottom row.

4.3 Matching in the *E. coli* genome:

1. Let f_i be the frequency of i-th nucleotide and compute $\sum {f_i}^2$, which for our data is $P = 0.250063$.
2. The probability of obtaining a match of length l is P^l. The expected number of matches of length l between two sequence of length L is $L^2 \cdot P^l$. In our case $L = 4639675$. If we make the assumption that the longest match of length l_{max} occurs only once, we can solve
$$1 = L^2 \cdot P^{l_{\max}}$$
for l_{max} to find
$$l_{max} = \frac{2 \cdot \ln L}{\ln(1/P)} \approx 22.1.$$
3. $100/(9+1) = 10$
4. $L \cdot P^{10} \approx 4.4$

4.6 Base your solution on a generalized suffix tree and consult Section 3.10.

4.7

1. It is a neuronal protein.
2. Yes, using the BLOSUM62 matrix (Table 2.4) we find that the score of alignment `EWL/KWL` is 16.
3. 59
4. $$S' = \frac{\lambda S - \ln K}{\ln 2} = \frac{0.267 \cdot 59 - \ln 0.0410}{\ln 2} = 27.335 \approx 27.$$
5. No, we expect approximately 10 such alignments by chance alone.
6. Score: no change as it is independent of database size; E-value: doubled, since $E = mn2^{-S'}$, i.e. if n is doubled while m and S' remain constant, E is doubled, too.

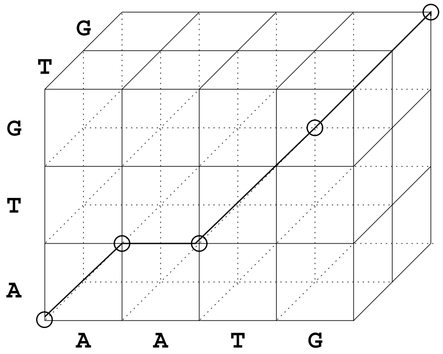
4.8

1. $S = 44$
2. $$S' = \frac{0.195 \cdot 44 - \ln 0.032}{\ln 2} = 17.3$$

Chapter 5

5.1 It implies that this position has been conserved throughout evolution and probably contributes critically to the 3D-structure of the globin protein.

5.3 The desired hyperlattice looks like this:



5.4 $(0, 0, 0) \rightarrow (1, 1, 0) \rightarrow (2, 1, 0) \rightarrow (3, 2, 1) \rightarrow (4, 3, 2)$

5.5 $S = -1 - 2 + 3 + 3 = 3$

5.6 Here is an alternative alignment:

```
AATG
-ATG
--TG
```

5.7 $(2, 2, 1, 1, 10, 1, 2)$

5.8 $(148{+}1){\cdot}(148{+}1){\cdot}(142{+}1){\cdot}(142{+}1){\cdot}(150{+}1){\cdot}(154{+}1){\cdot}(154{+}1) \approx 1.64{\cdot}10^{15}$

5.9

$$\frac{\log\left((2 \cdot 366 \cdot 24 \cdot 60 \cdot 60 + 8 \cdot 365 \cdot 24 \cdot 60 \cdot 60) \cdot 10^9\right)}{\log 101} \approx 8.76$$

5.11

$$\binom{n}{2} = \frac{n(n-1)}{2}$$

5.12 A protein sequence is a third as long as the corresponding DNA sequence. The pairwise and multiple alignment phases of the algorithm should therefore become nine times faster. The impact of this on overall performance depends on the relative amount of time taken by the alignment phases and construction of the neighbor joining tree ($O(n^3)$), as well as data loading, etc.

5.13 By following the alternative traceback path we get:

```
AACGT
A-CGT
A-AGT
A--GT
```

Chapter 6

6.1 The desired column in the profile looks like this:

Position	A	C	D	E	F	G	H	I	K	L	M	N	P	Q	R	S	T	V	W	Y
21	4	0	0	0	0	0	1	0	0	0	0	0	0	0	0	2	0	0	0	0

6.2 $P = 0.6^2 \cdot 0.4 \cdot 0.98^2 \approx 0.138.$

6.3 Illness: $1/0.4 = 2.5$; health: $1/0.02 = 50$.

6.4 Here is a possible result:

hidden	$\mathcal{H}$	$\mathcal{H}$	$\mathcal{H}$
observed	n	n	e

6.5 The next observation state in the sequence is `C`; given the HMM, we therefore get

$$\alpha_3(1) = 0.2 \cdot (0.14154 \cdot 0.98 + 0.02961 \cdot 0.07) \approx 0.028$$

$$\alpha_3(2) = 0.7 \cdot (0.14154 \cdot 0.02 + 0.02961 \cdot 0.93) \approx 0.021$$

6.6 The next symbol in the observation sequence is `A`; hence, we get the extended Viterbi matrix

	T	G	C	T	A
g_1	**0.7200**	← **0.1411**	←**0.0277**	←**0.0217**	←**0.0170**
g_2	0.0300	0.0195	0.0127	0.0035	0.0010

6.7 By writing the corresponding match states into columns we get the following alignment:

```
ATGA-
A-GA-
-TG-T
```

6.8 The induced alignment is ambiguous as the state sequences are compatible both with

```
ATGA
A-CA
```

and

```
ATGA
AC-A
```

Chapter 7

7.1 All three exons have reading frame 0.

7.2 The remainder of n exons can be computed as

$$\begin{aligned} &\text{remainder}(n) = 0 \\ &\text{remainder}(y-1) = (3 + \text{remainder}(y) - \text{length}(y)\%3)\%3 \end{aligned}$$

$$\begin{array}{r|ccc} \text{length}(y)\%3 \rightarrow & 0 & 1 & 2 \\ \text{remainder}(y) \quad \downarrow & & & \\ \hline 0 & 0 & 2 & 1 \\ 1 & 1 & 0 & 2 \\ 2 & 2 & 1 & 0 \end{array}$$

7.3 $\text{remainder} = (\text{length} - \text{frame})\%3$

7.4 Expect

$$E = 10^5 \cdot 0.25^2 = 6250$$

donor and acceptor dinucleotides each.

7.5 Recall that `TAA`, `TAG`, and `TGA` are stop codons. Hence we get for the various G/C-contents:

(a) $E = 3 \cdot (1/4)^3\, 2 \cdot 98 = 9.1875$
(b) $E = \left(2 \cdot (0.35)^2 \cdot 0.15 + (0.35)^3\right)\, 2 \cdot 98 = 15.6065$
(c) $E = \left(2 \cdot (0.15)^2 \cdot 0.35 + (0.15)^3\right)\, 2 \cdot 98 = 3.7485$

7.6 The average distance in an infinitely long sequence is the mean of the geometric distribution:

$$E = \sum_{i=0}^{\infty} i(15/16)^i(1/16) = 15$$

If sequences have a finite length n, then the upper bound for the exon length is $i_{\max} = n - 4$ (to accommodate both flanking dinucleotides). At least one acceptor dinucleotide has to be found among $n-(i+2)-1$ possible start positions. Therefore,

$$E = \sum_{i=0}^{n-4} i(15/16)^i(1/16)\left(1 - (15/16)^{n-(i+3)}\right).$$

For instance, for $n = 100$, one finds $E = 14.23$.

7.7 There are 4^9 possible nonamers. The number of nonamers with a score larger than $-\infty$ when scored with matrix from Figure 7.9 is

$$4 \cdot 3 \cdot 2 \cdot 1 \cdot 1 \cdot 3 \cdot 2 \cdot 2 \cdot 1 = 288\,.$$

Out of these, 287 nonamers have a score larger than 0. Therefore, one expects to find

$$E = 10^5 \frac{287}{4^9} \approx 109$$

donor sites with a positive score in a sequence of 100 kb. The expected number of `GT` dinucleotides is $10^5/16 = 6250$, which is about 57 times as many as the ones qualified as promising donor sites by the profile matrix.

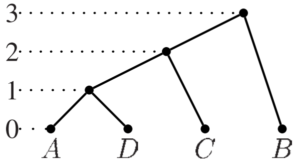
Chapter 8

8.1 Unrooted: 3; rooted: 15.

8.2 There are six distances between four taxa:

	A	B	C	D
A	-			
B	6	-		
C	4	6	-	
D	2	6	4	-

8.3 Here is the phylogeny:



1. The phylogeny is rooted: there is a node that denotes the ancestor of the entire clade.
2. The tree is ultrametric. Since the distances fit this tree exactly, they too are ultrametric.

8.4 No; a counter example obtained by permutating the entries in the distance matrix shown in Figure 8.13 would be

	1	2	3	4
1	-			
2	2	-		
3	6	2		
4	2	4	2	-

8.7 Notice that all polymorphic positions in the alignment except position 7 are singletons. These are not informative for maximum parsimony evaluation, as they add one mutation under any tree topology. The best tree under maximum parsimony can therefore simply be found by considering position 7; it is ((A,B)(C,D)) and its length $L(T) = 9$.

8.8 Remember that nearest neighbor interchange will generate $2(5-3) = 4$ trees, pruning and regrafting $4(5-3)(5-2) = 24$. The latter cannot all be distinct as there are only $3 \cdot 5 = 15$ topologies to choose from.

Chapter 9

9.2 See p. 432 in the population genetics textbook by Crow and Kimura [43].

9.5 $K(t) = -1/2 \log(1 - 2P - Q) - 1/4 \log(1 - 2Q)$, where P represent the percentage of transitions and Q that of transversions.

Chapter 10

10.2 Bacteria are haploid organisms. While we can still compute the probability that two genes drawn at random from the population are different, calling this quantity "heterozygosity" is strictly speaking a misnomer. Hence, the "virtual".

10.3 Solve

$$\frac{2 \cdot 10^{-8} \cdot N}{2 \cdot 10^{-8} \cdot N + 1} = 0.242$$

for N to find $N \approx 1.6 \cdot 10^7$.

10.4

$$H = \frac{2 \cdot 10^{10} \cdot 10^{-8}}{2 \cdot 10^{10} \cdot 10^{-8} + 1} \approx 0.995$$

10.5

$$H = \frac{2 \cdot 1.8 \cdot 10^{8} \cdot 10^{-8}}{2 \cdot 1.8 \cdot 10^{8} \cdot 10^{-8} + 1} \approx 0.783$$

10.7

$$\left(1 - \frac{1}{50}\right)^9 99 \cdot \frac{1}{50} \approx 3.4 \cdot 10^{-11}$$

10.8 Use Equation (10.25) and solve

$$10 = \sum_{i=0}^{10} \frac{\hat{\theta}}{\hat{\theta} + i}$$

for $\hat{\theta}$ to find $\hat{\theta} = 48.11$.

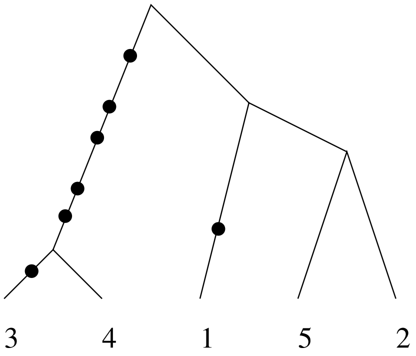
10.9 Homozygosity is a quadratic function in the allele frequencies, therefore there is only one global minimum. Show by induction on the number of alleles k that the minimum is attained at $x_i = 1/k$, for $1 \leq i \leq k$.

Chapter 12

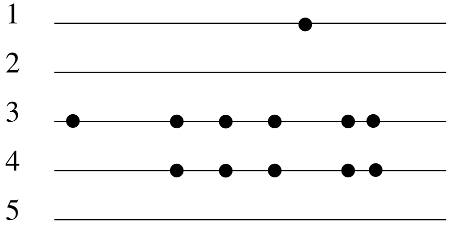
12.1 $H_4 = 2.08333$, $H_4^{(2)} = 1.42361$

12.2 There are n external and $n - 2$ internal branches.

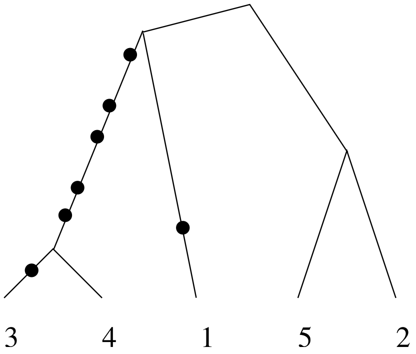
12.4 Here is one possible toplology:



12.5 The position of the groups of mutations along the sequences is entirely arbitrary. One of the infinitely many alternative arrangements might look like this:



12.6 Again, there is a large number of possibilities, for example:



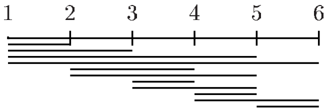
12.7 Reciprocal recombination would generate `AC` and `TG`.

12.8 The graphic below shows under **A** the matrix classifying all pairs of loci according to whether or not all four alleles are present; the corresponding list of intervals (**B**) is searched for overlaps, leaving four intervals (**C**). This means that $R_{\mathrm{M}} = 4$.

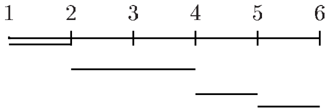
A

	1	2	3	4	5	6
1	-	1	1	0	1	1
2		-	0	1	1	0
3			-	1	1	0
4				-	1	1
5					-	1
6						-

B



C



12.9 $V_\mathrm{D} = 1.8667$; $V_\mathrm{e} = 0.8889$; $I_\mathrm{A}^\mathrm{S} = 0.3667$

Appendix B

B.1

$$\left(\frac{1}{10^4}\right)^2 = 10^{-8}.$$

B.2 The probability that you are ill given a positive test result is

$$\begin{aligned} P(\mathrm{ill}|\mathrm{test}^+) &= \frac{P(\mathrm{test}^+|\mathrm{ill})P(\mathrm{ill})}{P(\mathrm{test}^+, \mathrm{ill}) + P(\mathrm{test}^+, \mathrm{healthy})} \\ &= \frac{10^{-6}}{10^{-6} + 10^{-4}(1 - 10^{-6})} \\ &\approx 1\%. \end{aligned}$$

B.3 Multiply the genome length by the mutation rate:

$$120 \cdot 10^6 \cdot 10^{-8} = 1.2.$$

B.4 Assuming that mutations are Poisson distributed the probability of obtaining ≥ 1 mutations is the complement of obtaining no mutation:

$$P(\geq 1 \text{ mutations}) = 1 - e^{-120 \cdot 10^6 \cdot 10^{-8}} \approx 0.70.$$

References

1. M. Abramowitz and I. Stegun. *Handbook of Mathematical Functions*. Dover, New York, 1970.
2. J. F. Abril, R. Guigó, and T. Wiehe. `gff2aplot`: plotting sequence comparisons. *Bioinformatics*, 19:2477–2479, 2004.
3. M. D. Adams et al. The genome sequence of *Drosophila melanogaster*. *Science*, 287:2185–2195, 2000.
4. M. Aguadé, N. Miyashita, and C. H. Langley. Restriction map variation at the *zeste–tko* region in natural populations of *Drosophila melanogaster*. *Molecular Biology and Evolution*, 6:123–130, 1989.
5. A. Aho and M. Corasick. Efficient string matching: an aid to bibliographic search. *Communications of the ACM*, 18:333–340, 1975.
6. G. M. Air, A. R. Coulson, J. C. Fiddes, T. Friedmann, C. A. Hutchison III, F. Sanger, P. M. Slocombe, and A. J. H. Smith. Nucleotide sequence of the F protein coding region of bacteriophage FX174 and the amino acid sequence of its product. *Journal of Molecular Biology*, 125:247–254, 1978.
7. B. Alberts, A. Johnson, J. Lewis, M. Raff, K. Roberts, and P. Walter. *Molecular Biology of the Cell*. Garland, New York, 4th edition, 2002.
8. S. F. Altschul, M. S. Boguski, W. Gish, and J. C. Wootton. Issues in searching molecular sequence databases. *Nature Genetics*, 6:119–129, 1994.
9. S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman. Basic local alignment search tool. *Journal of Molecular Biology*, 215:403–410, 1990.
10. S. F. Altschul, T. L. Madden, A. A. Schäffer, J. Zhang, Z. Zhang, W. Miller, and D. Lipman. Gapped blast and psi-blast: a new generation of protein database search programs. *Nucleic Acids Research*, 25:3389–3402, 1997.
11. D. Altshuler, L. D. Brooks, A. Chakravarti, F. S. Collins, M. J. Daly, and P. Donnelly. A haplotype map of the human genome. *Nature*, 437:1299–1320, 2005.
12. C. F. Aquadro, K. M. Lado, and W. A. Noon. The *rosy* region of *Drosophila melanogaster* and *Drosophila simulans*. I. Contrasting levels of naturally occuring DNA restriction map variation and divergence. *Genetics*, 119:875–888, 1988.
13. O. T. Avery, C. M. MacLeod, and M. McCarty. Studies on the chemical nature of the substance inducing transformation of pneumococcal types. I. Induction of transformation by a deoxyribonucleic acid fraction isolated from *Pneumococcus* III. *Journal of Experimental Medicine*, 79:137–158, 1944.

14. E. Baake and H. Wagner. Mutation-selection models solved exactly with methods of statistical mechanics. *Genetical Research, Cambridge*, 78:93–117, 2001.
15. R. A. Baeza-Yates and C. H. Perleberg. Fast and practical approximate string matching. In *Proc. 3rd Symp. on Combinatorial Pattern Matching*, volume 644 of *Springer Lecture Notes in Computer Science*, pages 185–192, New York, 1992. Springer.
16. W. C. Barker and M. O. Dayhoff. Viral *src* gene products are related to the catalytic chain of mammalian cAMP-dependent protein kinase. *Proceedings of the American National Academy of Sciences, USA*, 79:2836–2839, 1982.
17. A. Bateman, L. Coin, R. Durbin, R. D. Finn, V. Hollich, S. Griffiths-Jones, A. Khanna, M. Marshall, S. Moxon, E. L. L. Sonnhammer, D. J. Studholme, C. Yeats, and S. R. Eddy. The PFAM protein families database. *Nucleic Acids Research*, 32:D138–D141, 2004.
18. J. Bedell, I. Korf, and M. Yandell. *BLAST. Basic Local Alignment Search Tool*. O'Reilly, 2003.
19. D.J. Begun and C. F. Aquadro. Molecular population genetics of the distal portion of the X chromosome in *Drosophila*: Evidence for genetic hitchhiking of the *yellow–achaete* region. *Genetics*, 129:1147–1158, 1991.
20. D.J. Begun and C. F. Aquadro. Levels of naturally occuring DNA polymorphism are correlated with recombination rates in *Drosophila melanogaster*. *Nature*, 356:519–520, 1992.
21. A. J. Berry, J. W. Ajioka, and M. Kreitman. Lack of polymorphism on the *Drosophila* fourth chromosome resulting from selection. *Genetics*, 129:1111–1117, 1991.
22. E. Birney and R. Durbin. Using GeneWise in the *Drosophila* annotation experiment. *Genome Research*, 10:547–548, 2000.
23. M. J. Bishop. Retroviruses and oncogenes II. In J. Lindsten, editor, *Nobel Lectures Physiology or Medicine*, pages 504–522. World Scientific, Singapore, 1989.
24. A. Bock, K. Forchhammer, J. Heider, W. Leinfelder, G. Sawers, B. Veprek, and F. Zinoni. Selenocysteine: the 21st amino acid. *Molecular Microbiology*, 5:515–520, 1991.
25. B. Boeckmann, A. Bairoch, R. Apweiler, M.-C. Blatter, A. Estreicher, E. Gasteiger, M. J. Martin, K. Michoud, C. O'Donovan, I. Phan, S. Pilbout, and M. Schneider. The Swiss-Pro protein knowledge base and its supplement TrEMBL in 2003. *Nucleic Acids Research*, 31:365–370, 2003.
26. S. Brenner, 1990. Remark made in a popular lecture on the human genome project at Cambridge University.
27. R. J. Britten and D. E. Kohne. Repeated sequences in DNA. *Science*, 161:529–540, 1968.
28. L. Bromham and D. Penny. The modern molecular clock. *Nature Reviews Genetics*, 4:216–224, 2003.
29. A. H. D. Brown, M. W. Feldman, and E. Nevo. Multilocus structure of natural populations of *Hordeum spontaneum*. *Genetics*, 96:523–536, 1980.
30. C. B. Burge and S. Karlin. Prediction of complete gene structures in human genomic DNA. *Journal of Molecular Biology*, 268:78–94, 1997.
31. C. B. Burge and S. Karlin. Finding the genes in genomic DNA. *Current Opinion in Structural Biology*, 8:346–354, 1998.
32. R. Bürger. *The Mathematical Theory of Selection, Recombination, and Mutation*. John Wiley & Sons, Chichester, 2000.
33. M. Burset and R. Guigó. Evaluation of gene structure prediction programs. *Genomics*, 34:353–357, 1996.
34. M. Burset, A. Seledtsov, and V. V. Solovyev. SpliceDB: database of canonical and non-canonical mammalian splice sites. *Nucleid Acids Research*, 29:255–259, 2001.

35. H. Carrillo and D. Lipman. The multiple sequence alignment problem in biology. *SIAM Journal of Applied Mathematics*, 48:1073–1082, 1988.
36. A. Chakravarti. Population genetics—making sense out of sequence. *Nature Genetics*, 21:56–60, 1999.
37. J. T. Chang. Recent common ancestors of all present-day individuals. *Advances in Applied Probability*, 31:1002–1026, 1999.
38. B. Charlesworth, M. T. Morgan, and D. Charlesworth. The effect of deleterious mutations on neutral molecular variation. *Genetics*, 134:1289–1303, 1993.
39. F. S. Collins, E. D. Green, A. E. Guttmacher, and M. S. Guyer. A blueprint for the genomic era. *Nature*, 422:835–847, 2003.
40. Mouse Genome Sequencing Consortium. Initial sequencing and comparative analysis of the mouse genome. *Nature*, 420:520–561, 2002.
41. T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. The MIT Press, Cambridge, MA, 2001.
42. F. H. C. Crick. On protein synthesis. *Symposium of the Society of Experimental Biology*, 12:138–163, 1957.
43. J. F. Crow and M. Kimura. *An Introduction to Population Genetics Theory*. Harper & Row, London, 1970.
44. C. Darwin. *On the Origin of Species by Means of Natural Selection or the Preservation of Favoured Races in the Struggle for Life*. Penguin, London, Penguin Classics, 1985 edition, 1859.
45. R. Dawkins. *The Ancestor's Tale*. Phoenix, London, 2004.
46. M. O. Dayhoff, R. M. Schwartz, and B. C. Orcutt. A model of evolutionary change in proteins. In M. O. Dayhoff, editor, *Atlas of Protein Sequence and Structure*, volume 5/suppl.3, pages 345–352. National Biomedical Research Foundation, Washington DC, 1978.
47. S. de Chadarevian. *Designs for Life*. Cambridge University Press, Cambridge, 2002.
48. A. L. Delcher, S. Kasti, R. D. Fleischmann, J. Peterson, O. White, and S. L. Salzberg. Alignment of whole genomes. *Nucleic Acids Research*, 27:2369–2376, 1999.
49. R. E. Dickerson and I. Geis. *Hemoglobin: Structure, Function, Evolution, and Pathology*. Benjamin/Cummings Publishing Company, Menlo Park, CA, 1983.
50. R. F. Doolittle, M. W. Hunkapiller, L. E. Hood, S. G. Devare, K. C. Robbins, S. A. Aaronson, and H. N. Antoniades. Simian sarcoma virus onc gene, *v-sis*, is derived from the gene (or genes) encoding a platelet-derived growth factor. *Science*, 221:275–277, 1983.
51. J. Drake, B. Charlsworth, D. Charlsworth, and F. J. Crow. Rates of spontaneous mutation. *Genetics*, 148:1667–1686, 1998.
52. R. Durbin, S. Eddy, A. Krogh, and G. Mitchison. *Biological Sequence Analysis: Probabilistic Models of Protein and Nucleic Acids*. Cambridge University Press, Cambridge, 1998.
53. W. F. Eanes, J. W. Ajioka, J. Hey, and C. Wesley. Restriction map variation associated with the *G6pd* polymorphism in natural populations of *Drosophila melanogaster*. *Molecular Biology and Evolution*, 6:384–397, 1989.
54. S. R. Eddy. Profile hidden Markov models. *Bioinformatics*, 14:755–763, 1998.
55. A. W. F. Edwards and L. L. Cavalli-Sforza. Reconstruction of evolutionary trees. In V. H. Heywood and J. McNeill, editors, *Phenetic and Phylogenetic Classification*, pages 67–76. The Systematics Association Publ. No. 6, London, 1964.
56. B. Efron. Bootstrap methods: another look at the Jackknife. *The Annals of Statistics*, 7:1–26, 1979.

57. B. Efron and R. J. Tibshirani. *An Introduction to the Bootstrap*. Chapman & Hall, New York, 1993.
58. M. Eigen. Molekulare Selbstorganisation und Evolution (Self organization of matter and the evolution of biological molecules). *Naturwissenschaften*, 58:465–523, 1971.
59. M. Eigen. *Stufen zum Leben; Die frühe Evolution im Visier der Molekularbiologie*. Piper, München, 1987.
60. M. Eigen, R. Winkler-Oswatitsch, and A. Dress. Statistical geometry in sequence space: a method of quantitative comparative sequence analysis. *Proceedings of the National Academy of Sciences, USA*, 85:5913–5917, 1988.
61. S. F. Elena and R. E. Lenski. Evolution experiments with microorganisms: the dynamics and genetic bases of adaptation. *Nature Reviews Genetics*, 4:457–469, 2003.
62. A. Elofsson. A study on how to best align protein sequences. *Proteins: Structure, Function, Genetics*, 46:300–309, 2002.
63. W. Enard, M. Przeworski, W. E. Fisher, C. S. L. Lai, V. Wiebe, T. Kitano, A. P. Monaco, and S. Pääbo. Molecular evolution of *FOXP2*, a gene involved in speech and language. *Nature*, 418:869–872, 2002.
64. W. J. Ewens. The sampling theory of selectively neutral alleles. *Theoretical Population Biology*, 3:87–112, 1972.
65. W. J. Ewens. *Mathematical Population Genetics I. Theoretical Introduction*. Springer, Heidelberg, 2nd edition, 2004.
66. W. J. Ewens and G. R. Grant. *Statistical Methods in Bioinformatics*. Springer, Heidelberg, 2001.
67. J. S. Farris. Estimating phylogenetic trees from distance matrices. *American Naturalist*, 106:645–668, 1972.
68. W. Feller. *An Introduction to Probability Theory and its Applications*, volume vol 1, 2nd edition. Wiley, New York, 1957.
69. J. Felsenstein. Cases in which parsimony or compatibility methods will be positively misleading. *Systematic Zoology*, 27:401–410, 1978.
70. J. Felsenstein. Evolutionary trees from DNA sequences: a maximum likelihood approach. *Journal of Molecular Evolution*, 17:368–376, 1981.
71. J. Felsenstein. Confidence limits on phylogenies: an approach using the bootstrap. *Evolution*, 39:783–791, 1985.
72. J. Felsenstein. PHYLIP (phylogeny interference package), 1993.
73. J. Felsenstein. *Inferring Phylogenies*. Sinauer, Sunderland, 2004.
74. D.-F. Feng and R. F. Doolittle. Progressive sequence alignment as a prerequisite to correct phylogenetic trees. *Journal of Molecular Evolution*, 25:351–360, 1987.
75. J. W. Fickett. Recognition of protein coding regions in DNA sequences. *Nucleic Acids Research*, 10:5303–5318, 1982.
76. R. A. Fisher. *The Genetical Theory of Natural Selection*. Oxford University Press, Oxford, Variorum edition, 1930/1999.
77. W. Fitch. Toward defining the course of evolution: minimum change for a specific tree topology. *Systematic Zoology*, 20:406–416, 1971.
78. R. D. Fleischmann et al. Whole-genome random sequencing and assembly of *Haemophilus influenzae* Rd. *Science*, 269:496–512, 1995.
79. L. Florea, G. Hartzell, Z. Zhang, G. M. Rubin, and W. Miller. A computer program for aligning a cDNA sequence with a genomic DNA sequence. *Genome Research*, 8:967–974, 1998.
80. J. E. F. Friedl. *Mastering Regular Expressions*. O'Reilly, Sebastopol, CA, 1997.
81. Y.-X. Fu and W.-H. Li. Statistical tests of neutrality of mutations. *Genetics*, 133:693–709, 1993.

82. P. A. Fuerst, R. Chakraborty, and M. Nei. Statistical studies on protein polymorphism in natural populations. I. Distribution of single locus heterozygosity. *Genetics*, 86:455–483, 1977.
83. T. Gaasterland, A. Sczyrba, E. Thomas, G. Aytekin-Kurban, P. Gordon, and C. W. Sensen. MAGPIE/EGRET annotation of the 2.9-mb *Drosophila melanogaster Adh* region. *Genome Research*, 10:502–510, 2000.
84. A. Y. Game and J. G. Oakeshott. The association between restriction site polymorphism and enzyme activity variation for *Esterase6* in *Drosophila melanogaster*. *Genetics*, 126:1021–1031, 1990.
85. M. S. Gelfand, A. A. Mironov, and P. A. Pevzner. Gene recognition *via* spliced alignment. *Proceedings of the National Academy of Sciences, USA*, 93:9061–9066, 1996.
86. Genetics_Computer_Group. Program manual for the Wisconsin package, version 8, 1994.
87. J. H. Gillespie. *Population Genetics*. Johns Hopkins University Press, Baltimore, 1998.
88. A. Goffeau et al. Life with 6000 genes. *Science*, 274:546–567, 1996.
89. M. Gribskov and R. R. Burgess. Sigma factors from *E. coli*, *B. subtilis*, phage SPO1, and phage T4 are homologous proteins. *Nucleic Acids Research*, 14:6745–6763, 1986.
90. M. Gribskov, A. D. McLachlan, and D. Eisenberg. Profile analysis: Detection of distantly related proteins. *Proceedings of the National Academy of Sciences, USA*, 84:4355–4358, 1987.
91. F. Griffith. The significance of pneumococcal types. *Journal of Hygiene*, 27:113–159, 1928.
92. R. C. Griffiths and P. Marjoram. An ancestral recombination graph. In P. Donnelly and S. Tavaré, editors, *Progress in Population Genetics and Human Evolution*, volume 87, pages 257–270. Springer, Berlin, 1997.
93. Z. Gu, L. M. Steinmetz, X. Gu, C. Scharfe, R. W. Davis, and W.-H. Li. Role of duplicate genes in genetic robustness against null mutations. *Nature*, 421:63–66, 2003.
94. R. Guigó. Computational gene identification. *Journal of Molecular Medicine*, 75:389–393, 1997.
95. R. Guigó. Assembling genes from predicted exons in linear time with dynamic programming. *Journal of Computational Biology*, 5:681–702, 1998.
96. R. Guigó, S. Knudsen, N. Drake, and T. Smith. Prediction of gene structure. *Journal of Molecular Biology*, 226:141–157, 1992.
97. R. Guigó and T. Wiehe. Gene prediction accuracy in large DNA sequences. In M. T. Galperin and E. V. Koonin, editors, *Frontiers in Computational Genomics*, pages 1–33. Caister Academic Press, Norwich, 2002.
98. S. K. Gupta, J. D. Kececioglu, and A. A. Schäffer. Improving the practical space and time efficiency of the shortest-path approach to sum-of-pairs multiple sequence alignment. *Journal of Computational Biology*, 2:459–472, 1995.
99. D. Gusfield. *Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology*. Cambridge University Press, Cambridge, 1997.
100. J. B. S. Haldane. A mathematical theory of natural and artificial selection. Part V: Selection and mutation. *Proceedings of the Cambrige Philosophical Society*, 23:838–844, 1928.
101. J. B. S. Haldane. *The Causes of Evolution*. Longmans and Green, London, 1932.
102. J. B. S. Haldane. The cost of natural selection. *Genetics*, 55:511–524, 1957.
103. B. Harr, M. Kauer, and C. Schlötterer. Hitchhiking mapping: a population-based fine-mapping strategy for adaptive mutations in *Drosophila melanogaster*. *Proceedings of the National Academy of Sciences, USA*, 99:12949–12954, 2002.

104. H. Harris. Enzyme polymorphism in man. *Proceedings of the Royal Society, London, B*, 164:298–310, 1966.
105. N. L. Harris and P. Senapathy. Distribution and consensus of branch point signals in eukaryotic genes: a computerized statistical analysis. *Nucleic Acids Research*, 18:3015–3019, 1990.
106. D. L. Hartl and A. G. Clark. *Principles of Population Genetics*. Sinauer, Sunderland, 1997.
107. D. L. Hartl, E. N. Moriyama, and S. A. Sawyer. Selection intensity for codon bias. *Genetics*, 138:227–234, 1994.
108. B. Haubold and R. R. Hudson. Lian 3.0: detecting linkage disequilibrium in multilocus data. *Bioinformatics*, 16:847–848, 2000.
109. B. Haubold, M. Travisano, P. B. Rainey, and R. R. Hudson. Detecting linkage disequilibrium in bacterial populations. *Genetics*, 150:1341–1348, 1998.
110. B. Haubold and T. Wiehe. Comparative genomics: methods and applications. *Naturwissenschaften*, 91:405–421, 2004.
111. K. Hayasaka, T. Gojobori, and S. Horai. Molecular phylogeny and evolution of primate mitochondrial DNA. *Molecular Biology and Evolution*, 5:626–644, 1988.
112. J. Hein, M. Shierup, and C. Wiuf. *Gene Genealogies, Variation and Evolution: A primer in coalescent theory*. Oxford University Press, Oxford, 2004.
113. J. G. Henikoff and S. Henikoff. Blocks database and its applications. *Methods in Enzymology*, 266:88–105, 1996.
114. S. Henikoff and J. G. Henikoff. Amino acid substitution matrices from protein blocks. *Proceedings of the National Academy of Sciences, USA*, 89:10915–10919, 1992.
115. J. Hey. The neutralist, the fly and the selctionist. *Trends in Ecology and Evolution*, 14:35–37, 1999.
116. D. S. Hirschberg. A linear space algorithm for computing longest common subsequences. *Communications of the ACM*, 18:341–343, 1975.
117. A. Hodges. *Alan Turing, the Enigma*. Vintage, London, 1992.
118. J. L. Hubby and R. C. Lewontin. A molecular approach to the study of genic heterozygosity in natural populations. I. The number of alleles at different loci in *Drosophila pseudoobscura*. *Genetics*, 54:577–594, 1966.
119. R. R. Hudson. Properties of a neutral allele model with intragenic recombination. *Theoretical Population Biology*, 23:183–201, 1983.
120. R. R. Hudson. Gene genealogies and the coalescent process. *Oxford Surveys in Evolutionary Biology*, 7:1–44, 1990.
121. R. R. Hudson. Analytical results concerning linkage disequilibrium in models with genetic transformation and conjugation. *Journal of Evolutionary Biology*, 7:535–548, 1994.
122. R. R. Hudson. Generating samples under a Wright-Fisher neutral model of genetic variation. *Bioinformatics*, 18:337–338, 2002.
123. R. R. Hudson and N. L. Kaplan. Statistical properties of the number of recombination events in the history of a sample of DNA sequences. *Genetics*, 111:147–164, 1985.
124. R. R. Hudson, M. Kreitman, and M. Aguadé. A test of neutral molecular evolution based on nucleotide data. *Genetics*, 116:153–159, 1987.
125. J. P. Huelsenbeck and B. Rannala. Phylogenetic methods come of age: testing hypotheses in an evolutionary context. *Science*, 276:227–232, 1997.
126. A. J. Iafrate, L. Feuk, M. N. Rivera, M. L. Listewnik, P. K. Donahoe, Y. Qi, S. W. Scherer, and C. Lee. Detection of large-scale variation in the human genome. *Nature Genetics*, 36:949–951, 2004.

127. I. Ina. New methods for estimating the numbers of synonymous and nonsynonymous substitutions. *Journal of Molecular Evolution*, 40:190–226, 1995.
128. International Human Genome Sequencing Consortium. Initial sequencing and analysis of the human genome. *Nature*, 409:860–921, 2001.
129. H. F. Judson. *The Eighth Day of Creation*. Penguin Books, London, 1995.
130. T. H. Jukes and C. R. Cantor. Evolution of protein molecules. In H. N. Munro, editor, *Mammalian Protein Metabolism*, volume 3, pages 21–132. Academic Press, New York, 1969.
131. N. Kaplan and R. R. Hudson. The use of sample genealogies for studying a selectively neutral m-loci model with recombination. *Theoretical Population Biology*, 28:382–396, 1985.
132. N. L. Kaplan, R. R. Hudson, and C. H. Langley. The ”hitchhiking effect” revisited. *Genetics*, 123:887–899, 1989.
133. O. H. Kapp, L. Moens, J. Vanfleteren, C. N. A. Trothman, T. Suzuki, and S. N. Vinogradov. Alignment of 700 globin sequences: Extent of amino acid substitution and its correlation with variation in volume. *Protein Science*, 4:2179–2190, 1995.
134. S. Karlin and S. F. Altschul. Methods for assessing the statistical significance of molecular sequence features by using general scoring schemes. *Proceedings of the National Academy of Sciences, USA*, 87:2264–2268, 1990.
135. S. Karlin and H. M. Taylor. *A First Course in Stochastic Processes*. Academic Press, San Diego, 2nd edition, 1975.
136. S. Karlin and H. M. Taylor. *A First Course in Stochastic Processes*. Academic Press, San Diego, 1975.
137. J. C. Kendrew, R. E. Dickerson, B. E. Strandberg, R. G. Hart, and D. R. Davies. Structure of myoglobin: a three-dimensional Fourier synthesis at 2 Å resolution. *Nature*, 185:422–427, 1960.
138. F. Khadem-Vargha, D. G. Gadian, A. Copp, and M. Mishkin. *FOXP2* and the neuroanatomy of speech and language. *Nature Reviews Neuroscience*, 6:131–138, 2005.
139. M. Kimura. Evolutionary rate at the molecular level. *Nature*, 217:624–626, 1968.
140. M. Kimura. The neutral theory as a basis for understanding the mechanism of evolution and variation at the molecular level. In M. Kimura, editor, *Molecular Evolution, Protein Polymorphism and the Neutral Theory*, pages 3–56. Springer Verlag, Berlin, 1982.
141. M. Kimura. *The Neutral Theory of Molecular Evolution*. Cambridge University Press, Cambridge, 1983.
142. M. Kimura and J. F. Crow. The number of alleles that can be maintained in a finite population. *Genetics*, 49:725–738, 1964.
143. M. Kimura and T. Ohta. The average number of generations until extinction of an individual mutant gene in a finite population. *Genetics*, 63:701–709, 1969.
144. M. Kimura and M. Otha. Protein polymorphism as a phase of molecular evolution. *Nature*, 229:467–469, 1971.
145. J. L. King and T. H. Jukes. Non-Darwinian evolution. *Science*, 164:788–798, 1969.
146. L. King, 2005. Popular US talk-show host Larry King interrogates philosopher Barbara Forrest on intelligent design. Quoted in *Nature*, vol. 437, p. 12.
147. J. F. C. Kingman. The coalescent. *Stochastic Processes and their Applications*, 13:235–248, 1982.
148. J. F. C. Kingman. On the genealogy of large populations. *Journal of Applied Probability*, 19A:27–43, 1982.
149. J. F. C. Kingman. Origins of the coalescent: 1974–1982. *Genetics*, 154:1461–1463, 2000.

150. R. D. Knight, S. J. Freeland, and L. F. Landweber. Rewiring the keyboard: evolvability of the genetic code. *Nature Reviews Genetics*, 2:49–58, 2001.
151. D. E. Knuth. *The Art of Computer Programming*, volume 1. Addision Wesley, Boston, 1997.
152. M. Kreitman. Nucleotide polymorphism at the alcohol dehydrogenase locus of *Drosophila melanogaster*. *Nature*, 304:412–417, 1983.
153. M. Kreitman. The neutral theory is dead. Long live the neutral theory. *BioEssays*, 18:678–683, 1996.
154. A. Krogh. Using database matches with HMMGene for automated gene detection in *Drosophila*. *Genome Research*, 10:523–528, 2000.
155. A. Krogh, M. Brown, I. S. Mian, K. Sjölander, and D. Hausser. Hidden Markov models in computational biology. *Journal of Molecular Biology*, 235:1501–1531, 1994.
156. G. V. Kryukov, S. Castellano, S. V. Novoselov, A. V. Lobanov, O. Zehtab, R. Guigó, and V. N. Gladyshev. Characterization of mammalian selenoproteomes. *Science*, 300:1439–1443, 2003.
157. D. Kulp, D. Haussler, M. G. Reese, and F. H. Eeckman. A generalized hidden Markov model for the recognition of human genes in DNA. In *ISMB-96: Proceedings of the Fourth International Conference on Intelligent Systems for Molecular Biology*, pages 134–141. AAAI Press, Menlo Park CA, 1996.
158. S. Kurtz. Reducing the space requirement of suffix trees. *Software - Practice and Experience*, 29:1149–1171, 1999.
159. B. W. Lange, C. H. Langley, and W. Stephan. Molecular evolution of *Drosophila metallothionein* genes. *Genetics*, 126:921–932, 1990.
160. C. H. Langley. The molecular population genetics of *Drosophila*. In N. Takahata and J. F. Crow, editors, *Population Biology of Genes and Molecules*, pages 75–91. National Academy of Sciences, Baifukan, Japan, 1990.
161. C. H. Langley, E. A. Montgomery, and W. F. Quattlebaum. Restriction map variation in the *Adh* region of *Drosophila*. *Proceedings of the National Academy of Sciences, USA*, 79:5631–5635, 1982.
162. C. H. Langley, A. E. Shrimpton, T. Yamazaki, N. Miyashita, Y. Matsuo, and C. F. Aquadro. Naturally occuring variation in the restriction map of the *Amy* region of *Drosophila melanogaster*. *Genetics*, 119:619–629, 1988.
163. A. J. Leigh-Brown. Variation of the 87*A* heat shock locus in *Drosophila melanogaster*. *Proceedings of the National Academy of Sciences, USA*, 80:5350–5354, 1983.
164. B. R. Levin. Periodic selection, infectious gene exchange and the genetic structure of *E. coli* populations. *Genetics*, 99:1–23, 1981.
165. R. C. Lewontin. An estimate of average heterozygosity in man. *American Journal of Human Genetics*, 19:681–685, 1967.
166. R. C. Lewontin. *The Genetic Basis of Evolutionary Change*. Columbia University Press, New York, 1974.
167. R. C. Lewontin and J. L. Hubby. A molecular approach to the study of genic heterozygosity in natural populations. II. Amount of variation and degree of heterozygosity in natural populations of *Drosophila pseudoobscura*. *Genetics*, 54:595–609, 1966.
168. W.-H. Li. *Molecular Evolution*. Sinauer, Sunderland, 1997.
169. D. J. Lipman, S. F. Altschul, and J. D. Kececioglu. A tool for multiple sequence alignment. *Proceedings of the National Academy of Sciences, USA*, 86:4412–4415, 1989.
170. M. Lynch and J. S. Conery. The evolutionary fate and consequences of duplicate genes. *Science*, 290:1151–1155, 2000.

171. K. Makino, K. Oshima, K. Kurakowa, K. Yokoyama, T. Uda, K. Tagomori, Y. Iijima, M. Najima, M. Nakano, A. Yamashita, Y. Kubota, S. Kimura, T. Tasunga, T. Honda, H. Shinagawa, M. Hattori, and Y. Iida. Genome sequence of *Vibrio parahaemolyticus*: a pathogenic mechanism distinct from that of *V. cholerae*. *The Lancet*, 361:743–749, 2003.
172. U. Manber and E. W. Myers. Suffix arrays: a new method for on-line string searches. *SIAM Journal on Computing*, 22:935–948, 1993.
173. J. Maynard Smith. Natural selection and the concept of a protein space. *Nature*, 225:563–564, 1970.
174. J. Maynard Smith and J. Haigh. The hitch-hiking effect of a favourable gene. *Genetical Research, Cambridge*, 23:23–35, 1974.
175. J. Maynard Smith, N. H. Smith, C. G. Dowson, and B. G. Spratt. How clonal are bacteria? *Proceedings of the National Academy of Sciences, USA*, 90:4384–4388, 1993.
176. M. A. McClure, T. K. Vasi, and W. M. Fitch. Comparative analysis of multiple protein-sequence alignment methods. *Molecular Biology and Evolution*, 11:571–592, 1994.
177. E. M. McCreight. A space-economic suffix tree construction algorithm. *Journal of the ACM*, 23:262–272, 1976.
178. J. H. McDonald and M. Kreitman. Adaptive protein evolution at the *Adh* locus in *Drosophila*. *Nature*, 351:652–654, 1991.
179. E. W. Meyers. An overview of sequence comparison algorithms in molecular biology. Technical report, The University of Arizona, 1991.
180. E. W. Meyers et al. A whole-genome assembly of *Drosophila*. *Science*, 287:2196–2204, 2000.
181. E. W. Meyers and W. Miller. Optimal alignments in linear space. *Bioinformatics*, 4:11–17, 1988.
182. R. Milkman. Electrophoretic variation in *Escherichia coli* from natural sources. *Science*, 182:1024–1026, 1973.
183. N. Miyashita and C. H. Langley. Molecular and phenotypic variation of the *white* locus region in *Drosophila melanogaster*. *Genetics*, 120:199–212, 1988.
184. B. Morgenstern, A. Dress, and T. Werner. Multiple DNA and protein sequence alignment based on segment-to-segment comparison. *Proceedings of the National Academy of Sciences, USA*, 93:12098–12103, 1996.
185. R. Mott. EST_GENOME: a program to align spliced DNA sequences to unspliced genomic DNA. *Computer Applications in the Biosciences*, 13:477–478, 1997.
186. S. V. Muse and B. S. Weir. Testing for equality of evolutionary rates. *Genetics*, 132:269–276, 1992.
187. E. W. Myers and W. Miller. Optimal alignments in linear space. *Computer Applications in the Biosciences*, 4:11–17, 1988.
188. S. Nair, J. T. Williams, A. Brockman, L. Paiphun, Mayxay M., P. N. Newton, J.-P. Guthmann, F. M. Smithuis, T. T. Hien, N. J. White, F. Nosten, and T. J. C. Anderson. A selective sweep driven by pyrimethamine treatment in SE Asian malaria parasites. *Molecular Biology and Evolution*, Advance Access published June 27, 2003, 2003.
189. S. B. Needleman and C. D. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology*, 48:443–453, 1970.
190. R. Nielsen, C. Bustamante, A. G. Clark, S. Glanowski, T. B. Sackton, M. J. Hubisz, A. Fledel-Alon, D. M. Tanenbaum, D. Civello, T. J. White, J. J. Sninsky, M. Adams, and M. Cargill. A scan for positively selected genes in the genomes of humans and chimpanzees. *PLOSBiology*, 3:0976–0985, 2005.

191. M. W. Nierenberg and J. H. Matthaei. The dependence of cell-free protein synthesis in *E. coli* upon naturally occurring or synthetic polyribonucleotides. *Proceedings of the National Academy of Sciences, USA*, 47:1588–1602, 1961.
192. M. Nordborg. Coalescent theory. In D. J. Balding, M. Bishop, and C. Cannings, editors, *Handbook of Statistical Genetics*, chapter 7, pages 178–212. Wiley, New York, 2001.
193. L. Pachter, M. Alexandersson, and S. Cawley. Applications of generalized pair Hidden Markov Models to alignment and gene finding problems. In *Proceedings of the Fifth Annual Conference on Computational Molecular Biology (RECOMB 2001)*, pages 241–248. ACM Press, New York, 2001.
194. R. D. M. Page and R. C. Holmes. *Molecular Evolution: A Phylogenetic Approach*. Blackwell Science, 1998.
195. P. Pamilo and N. O. Bianchi. Evolution of the *Zfx* and *Zfy* genes: Rates and interdependence between the genes. *Molecular Biolology and Evolution*, 10:271–281, 1993.
196. G. Parra, P. Agarwal, J. F. Abril, T. Wiehe, J. Fickett, and R. Guigó. Comparative gene prediction in human and mouse. *Genome Research*, 13:108–117, 2003.
197. W. R. Pearson and D. J. Lipman. Improved tools for biological sequence comparison. *Proceedings of the National Academy of Sciences, USA*, 85:2444–2448, 1988.
198. W. R. Pearson and T. C. Wood. Statistical significance in biological sequence comparison. In D. J. Balding, M. Bishop, and C. Cannings, editors, *Handbook of Statistical Genetics*, chapter 2, pages 39–65. Wiley, New York, 2001.
199. H. Peled-Zehavi, J. A. Berglund, M. Rosbash, and A. D. Frankel. Recognition of RNA branch point sequences by the KH domain of splicing factor 1 (mammalian branch point binding protein) in a splicing factor complex. *Molecular Cell Biology*, 21:5232–5241, 2001.
200. M. Perutz. *I wish I'd made you angry earlier; Essays on Science and Scientists*. Oxford University Press, Oxford, 1998.
201. D. A. Petrov. Evolution of genome size: new approaches to an old problem. *Trends in Genetics*, 17:23–28, 2001.
202. S. Pietrokovski, J. G. Henikoff, and S. Henikoff. The blocks database—a system for protein classification. *Nucleic Acids Research*, 24:197–200, 1996.
203. S. Pinker. *The Language Instinct; The New Science of Language and Mind*. Allen Lane, London, 1994.
204. W. B. Provine. *Sewall Wright and Evolutionary Biology*. The University of Chicago Press, Chicago, 1986.
205. L. R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77:257–286, 1989.
206. M. G. Reese, G. Hartzell, N. L. Harris, U. Ohler, J. F. Abril, and S. E. Lewis. Genome annotation assessment in *Drosophila melanogaster*. *Genome Research*, 10:483–501, 2000.
207. M. G. Reese, D. Kulp, H. Tammana, and D. Haussler. Genie—Gene Finding in *Drosophila melanogaster*. *Genome Research*, 10:529–538, 2000.
208. D. L. T. Rhode, S. Olson, and J. T. Chang. Modelling the recent common ancestry of all living humans. *Nature*, 431:562–566, 2004.
209. P. Rice and A. Bleasby. EMBOSS: The European Molecular Biology Open Software Suite. *Trends in Genetics*, 16:276–277, 2000.
210. B. J. Richardson, P. R. Baverstock, and M. Adams. *Allozyme electrophoresis; a handbook for animal systematics and population studies*. Academic Press, London, 1986.
211. J. Rozas, J. C. Sánchez-DelBarrio, X. Messeguer, and R. Rozas. DnaSP, DNA polymorphism analyses by the coalescent and other methods. *Bioinformatics*, 19:2496–2497, 2003.

212. N. Saitou and M. Nei. The neighbor-joining method: a new method for reconstructing phylgenetic trees. *Molecular Biology and Evolution*, 4:406–425, 1987.
213. A. A. Salamov and V. V. Solovyev. *Ab initio* gene finding in *Drosophila* genomic DNA. *Genome Research*, 10:516–522, 2000.
214. F. Sanger, A. R. Coulson, T. Friedmann, G. M. Air, B. G. Barrell, N. L. Brown, J. C. Fiddes, C. A. Hutchison III, P. M. Slocombe, and M. Smith. The nucleotide sequence of bacteriophage ϕX174. *Journal of Molecular Biology*, 125:225–246, 1978.
215. F. Sanger, A. R. Coulson, G. F. Hong, D. F. Hill, and G. B. Petersen. Nucleotide sequence of bacteriophage λ DNA. *Journal of Molecular Biology*, 162:729–773, 1982.
216. S. W. Schaeffer, C. F. Aquadro, and C. H. Langley. Restriction map variation in the *Notch* region of *Drosophila melanogaster*. *Molecular Biology and Evolution*, 5:30–40, 1988.
217. C. Schlötterer. Hitchhiking mapping—functional genomics from the population genetics perspective. *Trends in Genetics*, 19:32–38, 2003.
218. H. A. Schmidt, K. Strimmer, M. Vingron, and A. von Haeseler. TREE-PUZZLE: maximum likelihood phylogenetic analysis using quartets and parallel computing. *Bioinformatics*, 18:502–504, 2002.
219. T. D. Schneider and R. M. Stephens. Sequence logos: A new way to display consensus sequences. *Nucleic Acids Research*, 18:6097–6100, 1990.
220. E. Schrödinger. *What is Life*. Canto. Cambridge University Press, Cambridge, 1944/1992.
221. E. A. Schultes and D. P. Bartel. One sequence, two ribozymes: Implications for the emergence of new ribozyme folds. *Science*, 289:448–452, 2000.
222. R. K. Selander and B. R. Levin. Genetic diversity and structure in *Escherichia coli* populations. *Science*, 210:545–547, 1980.
223. The Chimpanzee Sequencing and Analysis Consortium. Initial sequence of the chimpanzee genome and comparison with the human genome. *Nature*, 437:69–87, 2005.
224. C. E. Shannon. A mathematical theory of communication. *The Bell Systems Technical Journal*, 27:379–423, 1948.
225. T. F. Sharbel, B. Haubold, and T. Mitchell-Olds. Genetic isolation by distance in *Arabidopsis thaliana*: biogeography and postglacial colonization of Europe. *Molecular Ecology*, 9:2109–2118, 2000.
226. C. J. Sigrist, L. Cerutti, N. Hulo, A. Gattiker, L. Falquet, M. Pagni, A. Bairoch, and P. Bucher. PROSITE: a documented database using patterns and profiles as motif descriptors. *Briefings in Bioinformatics*, 3:265–274, 2002.
227. M. Slatkin and T. Wiehe. Genetic hitch-hiking in a subdivided population. *Genetical Research, Cambridge*, 71:155–160, 1998.
228. C. W. Smith and J. Valcarcel. Alternative pre-mRNA splicing: the logic of combinatorial control. *Trends in Biochemical Sciences*, 25:381–388, 2000.
229. R. F. Smith and T. F. Smith. Pattern-induced multi-sequence alignment (pima) algorithm employing secondary structure-dependent gap penalties for use in comparative protein modelling. *Protein Engineering*, 5:35–41, 1992.
230. T. F. Smith and M. S. Waterman. Identification of common molecular subsequences. *Journal of Molecular Biology*, 147:195–197, 1981.
231. R. R. Sokal and F. J. Rohlf. *Biometry*. W. H. Freeman & Company, New York, 2nd edition, 1981.
232. R. R. Sokal and P. H. A. Sneath. *Principles of Numerical Taxonomy*. W. H. Freeman & Company, New York, 1963.
233. E. L. L. Sonnhammer and R. Durbin. A dot-matrix program with dynamic threshold control suited for genomic DNA and protein sequence analysis. *Gene*, 167:1–10, 1996.

234. V. Souza, T. T. Nguyen, R. R. Hudson, D. Piñero, and R. E. Lenski. Hierarchical analysis of linkage disequilibrium in *Rhizobium* populations: evidence for sex? *Proceedings of the National Academy of Sciences, USA*, 89:8389–8393, 1992.
235. W. Stephan. Mathematical model of the hitchhiking effect, and its application to DNA polymorphism data. In O. Arino, D. E. Axelrod, M. Kimmel, and V. Capasso, editors, *Advances in Mathematical Dynamics - Molecules, Cells and Man*, page 2945. World Scientific, River Edge, NJ, 1997.
236. W. Stephan, T. Wiehe, and M. W. Lenz. The effect of strongly selected substitutions on neutral polymorphism: Analytical results based on diffusion theory. *Theoretical Population Biology*, 41:237–254, 1992.
237. K. Strimmer and A. v. Haeseler. Likelihood-mapping: a simple method to visualize phylogenetic content of a sequence alingment. *Proceedings of the National Academy of Sciences, USA*, 94:6815–6819, 1997.
238. D. L. Swofford, G. J. Olsen, P. J. Waddell, and D. M. Hillis. Phylogenetic inference. In D. M. Hillis, M. Craig, and B. K. Marble, editors, *Molecular Systematics*, pages 407–514. Sinauer, Sunderland, 2nd edition, 1996.
239. F. Tajima. Evolutionary relationship of DNA sequences in finite populations. *Genetics*, 105:437–460, 1983.
240. F. Tajima. Statistical method for testing the neutral mutation hypothesis by DNA polymorphism. *Genetics*, 123:585–595, 1989.
241. T. S. Takano, S. Kusakabe, and T. Mukai. The genetic structure of natural populations of *Drosophila melanogaster*. XXII. Comparative study of DNA polymorphisms in northern and southern natural populations. *Genetics*, 129:753–761, 1991.
242. D. Tautz. A genetic uncertainty problem. *Trends in Genetics*, 16:475–477, 2000.
243. The *C. elegans* Sequencing Consortium. Genome sequence of the nematode *C. elegans*: a platform for investigating biology. *Science*, 282:2012–2018, 1998.
244. The International SNP Map Working Group. A map of human genome sequence variation containing 1.42 million single nucleotide polymorphisms. *Nature*, 409:928–933, 2001.
245. The *Arabidopsis* Genome Initiative. Analysis of the genome sequence of the flowering plant *Arabidopsis thaliana*. *Nature*, 408:796–815, 2000.
246. J. D. Thompson, D. G. Higgins, and T. J. Gibson. CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position specific gap penalities and weight matrix choice. *Nucleic Acids Research*, 22:4673–4680, 1994.
247. J. D. Thompson, F. Plewniak, and O. Poch. A comprehensive comparison of multiple sequence alignment programs. *Nucleic Acids Research*, 27:2682–2690, 1999.
248. E. C. Uberbacher and R. J. Mural. Locating protein-coding regions in human DNA sequences by a multiple sensor–neural network approach. *Proceedings of the National Academy of Sciences, USA*, 88:11261–11265, 1991.
249. E. Ukkonen. On-line construction of suffix-trees. *Algorithmica*, 14:249–260, 1995.
250. J. Usuka and V. Brendel. Gene structure prediction by spliced alignment of genomic DNA with protein sequences: increased accuracy by differential splice site scoring. *Journal of Molecular Biology*, 297:1075–1085, 2000.
251. H. E. Varmus. Retroviruses and oncogenes I. In J. Lindsten, editor, *Nobel Lectures Physiology or Medicine*, pages 504–522. World Scientific, 1989.
252. J. C. Venter et al. The sequence of the human genome. *Science*, 291:1304–1351, 2001.
253. M. S. Waterman. *Introduction to Computational Biology; Maps, Sequences and Genomes*. Chapman & Hall/CRC, London, 1995.

254. J. D. Watson. *DNA — The Secret of Life*. Random House, New York, 2004.
255. J. D. Watson, T. A. Baker, S. P. Bell, A. Gann, M. Levine, and R. Losick. *Molecular Biology of the Gene*. Cold Spring Harbor Laboratory Press, Woodbury, NY, 2004.
256. J. D. Watson and F. H. C. Crick. Molecular structure of nucleic acids: a structure for deoxyribose nucleic acid. *Nature*, 171:737–738, 1953.
257. G. A. Watterson. Models for the logarithmic species abundance distributions. *Theoretical Population Biology*, 6:217–250, 1974.
258. G. A. Watterson. On the number of segregating sites in genetical models without recombination. *Theoretical Population Biology*, 7:256–276, 1975.
259. P. Weiner. Linear pattern matching algorithms. *Proceedings of the 14th IEEE Annual Symposium on Switching and Automata Theory*, pages 1–11, 1973.
260. T. Wiehe and W. Stephan. Analysis of a genetic hitchhiking model and its application to DNA polymorphism data from *Drosophila melanogaster*. *Molecular Biology and Evolution*, 10:842–854, 1993.
261. D. E. Wildman, M. Uddin, G. Liu, L. I. Grossman, and M. Goodman. Implications of natural selection in shaping 99.4% nonsynonymous DNA identity between humans and chimpanzees: enlarging genus *Homo*. *Proceedings of the National Academy of Sciences, USA*, 100:7181–7188, 2003.
262. S. Wright. Evolution in Mendelian populations. *Genetics*, 16:97–159, 1931.
263. S. Wright. The roles of mutation, inbreeding, crossbreeding and selection in evolution. *Proceedings of the Sixth International Congress of Genetics*, 1:356–366, 1932.
264. M. Q. Zhang. Computational prediction of eukaryotic protein-coding genes. *Nature Reviews Genetics*, 3:698–709, 2002.
265. E. Zuckerkandl and L. Pauling. Molecular disease, evolution and heterogeneity. In M. Kash and B. Pullman, editors, *Horizons in Biochemistry*, pages 189–225. Academic Press, New York, 1962.
266. E. Zuckerkandl and L. Pauling. Evolutionary divergence and convergence in proteins. In V. Bryson and H. J. Vogel, editors, *Evolving Genes and Proteins*, pages 97–166. Academic Press, New York, 1965.

Glossary

additive distances: Pairwise distances between taxa possibly evolving at different rates.

affine gap cost: Gap costs consisting of a constant part and a part that scales linearly with gap length: $G = g_\mathrm{o} + l \cdot g_\mathrm{e}$, where g_o is the gap opening cost, g_e the gap extension cost, and l the gap length.

alignment: An alignment consists of two or more sequences written on top of each other in such a way that either residues are paired with residues or with gaps representing insertions/deletions. The aim of alignment construction is usually to align homologous residues.

allele: A particular version of a gene.

analogy: Similarity between biological traits *not* based on common descent.

bioinformatics: Discipline at the interface between molecular biology and computer science established in the wake of the human genome project.

BLAST (Basic Local Alignment Search Tool): Software package for rapidly finding local alignments between a query (pattern) and a subject (text) sequence. In practice the subject often consists of a large set of sequences.

BLOSUM substitution matrices: Series of BLOcks SUbstitution Matrices containing scores for all possible pairs of amino acids. A matrix entry consists of the logarithm of the ratio between the probability of finding a pair of amino acids due to homology and the probability of finding it by chance alone. Hence, the scores are also known as log-odds scores. The BLOSUM series is based on blocks of protein sequences that can be aligned without gaps. Sequences with an identity of $\geq x$ percent are grouped together resulting in a BLOSUMx matrix.

bootstrap: Statistical procedure for simulating the process of repeatedly drawing samples from a population. This is done by drawing with replacement n elements from a sample of size n. When applied to phylogenetic analysis, the result consists of a phylogeny where numbers next to nodes indicate the frequency of obtaining the particular cluster in trees constructed from bootstrap samples of the input data.

breadth-first traversal: Tree traversal where sibling nodes are visited before child nodes.

cDNA: Complementary DNA. DNA molecule derived from a mRNA by reverse transcription.

centromere: Region in a chromosome where the spindle apparatus attaches during mitosis and meiosis. Often, but not always situated in the center of a chromosome and made up of repetitive, gene-poor sequence.

chromosome: Genomes are divided into one or more chromosomes, each consisting of one long DNA molecule and associated packaging proteins.

coalescence: The merging of two lineages in a coalescent when moving from the present into the past, i.e. from the tips of the coalescent toward its root.

coalescent: Gene genealogy used for the simulation of population genetic data.

codon: Triplet of nucleotides that codes for a single amino acid or stop.

DDBJ (DNA Data Bank of Japan): Comprehensive public nucleotide sequence database maintained by the Japanese National Institute of Genetics. Its function is similar to that of GenBank and EMBL.

depth-first traversal: Tree traversal where child nodes are visited before sibling nodes.

diploid: In a diploid organism such as humans each cell carries two copies of each chromosome.

disequilibrium mapping: Searching for genes by looking for associations between a phenotype and anonymous genetic markers. This marker should be in linkage disequilibrium with the desired gene and therefore close to it.

DNA (DeoxyriboNucleic Acid): Doublehelical molecule serving as repository of genetic information in the vast majority of organisms. By convention a given DNA sequence, e.g. `AGT`, is written in the 5’ to 3’ direction.

dotplot: A plot for comparing two sequences, each drawn along the edge of a two-dimensional matrix. Wherever the two sequences match, a dot is drawn.

dynamic programming: Computational method for solving recursive optimization problems by starting at the boundary conditions and storing subproblem solutions rather than recalculating them.

EMBL: European Molecular Biology Laboratory. Also the name of the comprehensive public nucleotide sequence database maintained by the European Bioinformatics Institute (EBI) in Cambridge, UK. Its function is similar to that of GenBank, and DDBJ.

EST (expressed sequence tag): A sequence tagged site derived from a cDNA.

eucaryote: Unicellular or multicellular organism where each cell contains a nucleus.

exact matching problem: The problem of finding in a text all the positions where an exact copy of a given pattern starts.

exon: Exons are the protein-coding regions of a primary transcript.

FASTA: Computer program for rapid local alignment; similar to BLAST.

fixation: Used in population genetics to describe the process whereby an allele becomes the only version of the corresponding gene in the population.

GenBank: Comprehensive public nucleotide sequence database maintained by the American National Center for Biotechnology Information (NCBI). Researchers reporting results based on new sequence data are usually required to deposit their sequence data either in GenBank, EMBL, or DDBJ before publishing their work.

gene: Transcribed stretch of DNA, possibly including its regulatory regions.

genetic diversity: The genetic diversity of a locus is the probability that two alleles of that locus randomly drawn from a population are different.

genetic drift: Random changes of allele frequencies in the course of evolution.

genetic marker: A genetic landmark of known position along a chromosome. Such a landmark might, for example, be a gene for white eye color in the fruit fly, or a sequenced stretch of DNA.

genome: The entirety of an organism's genetic information.

genomics: The study of whole genomes.

global alignment: An alignment strategy aimed at comparing the input sequences along their entire length. Based on the assumption that homology extends along the entire length of the sequences compared.

guide tree: In the context of multiple sequence alignment a guide tree is a binary tree with sequence designations at its leaves. Starting from the leaves and working toward the root, this tree specifies the order in which clusters of genes should be pairwise aligned in order to construct a multiple alignment.

haploid: In a haploid organism each cell carries just a single copy of each chromosome. Many asexual organisms, including most procaryotes are haploid.

hash table: Generalized array where any object, e.g. a string, can serve as index.

heterozygosity: The fraction of loci with distinct alleles in a diploid organism; more generally also used as a synonym of genetic diversity.

heuristic alignment: Alignment strategy using approximate procedures for obtaining an algorithm that is faster than optimal alignment algorithms. BLAST and FASTA are well-known examples of programs based on heuristic alignment procedures.

hidden Markov model: Stochastic model consisting of ≥ 2 hidden and ≥ 2 observable states. Hidden states can change into each other with probabilities also specified in the model. A given hidden state emits the observable states with characteristic and predefined probabilities. A typical application of hidden Markov models is gene prediction where the observable states are the nucleotides. The hidden states to be inferred from the observable states might include intron, initial exon, internal exon, final exon, intergenic region, etc.

hitchhiking: In the context of population genetics this refers to the process whereby genetic diversity linked to an advantageous allele "hitches a ride" to fixation.

homology: Similarity between biological traits based on common descent.

inexact matching problem: The problem of finding in a text T all the positions where an inexact copy of a given pattern P starts. Inexactness might be restricted to mismatches or might include insertions/deletions.

infinite alleles model: Model of evolution where each mutation at a locus generates a new allele. If a sequence of, say, 500 nucleotides is considered a locus and the sequence as its allele, then the infinite alleles model is likely to apply.

infinite sites model: Model of evolution where each mutation at a locus affects a different site. This models mutations in a DNA sequence of reasonable length, where the probability of two mutations affecting the same residue is negligible.

intron: Introns are regions of a primary transcript that are not protein coding; they are removed by splicing. Introns are common in eucaryotic genes but very rare in procaryotic genes.

keyword tree: Data structure for simultaneously searching a set of key words in a text.

linkage: Loci residing on the same chromosome are said to be linked.

linkage equilibrium: Loci that are statistically independent of each other are said to be in linkage equilibrium. This amounts to the requirement that the frequency of a genotype consisting of alleles at two or more loci is equal to the product of the alleles' frequencies.

linkage disequilibrium: The absence of linkage equilibrium. Due to reciprocal recombination, linkage disequilibrium between two loci decays with distance.

local alignment: An alignment strategy aimed at detecting homologous regions between the input sequences.

locus: A position on a chromosome, often synonymous with gene.

match: In the context of string searching two identical characters are said to form a match.

maximum likelihood: Framework for statistical inference. In the context of maximum likelihood phylogeny reconstruction it provides the criterion for evaluating phylogenies by looking for the tree that maximizes the likelihood of the observed data.

maximum parsimony: Criterion for evaluating phylogenies by minimizing the number of evolutionary changes (usually mutations) they imply.

microsatellite: Tandem repeats with short repeat units, e.g. $(\texttt{GC})_n$. Frequently used as genetic markers.

mismatch: In the context of string searching two distinct characters are said to form a mismatch.

molecular clock: Evolutionary scenario in which the rate of evolution along all lineages is equal.

molecular evolution: The study of evolution at the molecular level, i.e. by analyzing DNA and protein sequence data.

multiple alignment: Alignment between ≥ 2 sequences.

mutation: Random change in a protein or nucleotide sequence.

neighbor-joining: Method for reconstructing a phylogeny from additive pairwise distance data.

neutral evolution: Evolutionary change without fitness effect.

non-synonymous substitution: Substitution in a protein-coding DNA sequence that changes the encoded protein sequence.

open reading frame: Segment of a nucleotide sequence ranging from a start to a stop codon.

optimal alignment: Alignment method guaranteed to find the best possible alignment under the model. In practice this is usually based on dynamic programming.

ORF: Open Reading Frame.

orthology: Similarity between homologous biological traits carried by different organisms.

overlap alignment: Alignment strategy for overlapping sequences. Applied in the assembly of sequence fragments generated in the course of a shotgun sequencing project.

pairwise alignment: Alignment between two sequences.

PAM substitution matrices: Series of Percent Accepted Mutations matrices containing scores for all possible pairs of amino acids. A matrix entry consists of the logarithm of the ratio between the probability of finding a pair of amino acids due to homology and the probability of finding it by chance alone. Hence, the scores are also known as log-odds scores.

paralogy: Similarity between homologous biological traits carried by the same organism.

pattern: In the context of string searching this refers to the string searched for in a text.

PCR (Polymerase chain reaction): Procedure for rapidly increasing the concentration of a specific stretch of DNA sequence in a reaction mixture by orders of magnitude.

PDB (Protein Data Base): Public collection of three-dimensional protein structures.

phylogeny: Species genealogy, usually computed from empirical data.

polyploid: In a polyploid organism each cell carries multiple copies of each chromosome. For example, in hexaploid wheat each chromosome is present in six copies.

population genetics: Study of evolutionary forces acting at the level of biological populations.

profile: Also known as position-specific weight matrix. Used to represent protein families.

procaryote: Organism without a nucleus; all procaryotes are unicellular.

protein family: Group of proteins with similar functions.

protein: Macromolecule consisting of amino acids, the sequence of which determines its three-dimensional shape and, hence, its function. Their amino acid sequence is specified in their gene's DNA sequence. By convention a given protein sequence, e.g. `MTY`, is written in the N-terminal to C-terminal direction.

proteomics: The study of the entire set of proteins from one or more organisms.

reading frame: One of six possible ways in which a DNA sequence can in theory be translated by starting at positions 1, 2, or 3 on the forward strand or at positions 1, 2, or 3 on the reverse strand.

recombination: Exchange of genetic material between homologous chromosomes during crossing over in meiosis. Reciprocal recombination refers to the process whereby after recombination both participating chromosomes are changed downstream of the recombination point. In contrast, gene conversion leaves one of the participating chromosomes unchanged.

recursion: A function expressed as a function of itself; for example, the i-th Fibonacci number is expressed recursively as the sum of the $i-1$-th and the $i-2$-th Fibonacci number: $f_i = f_{i-1} + f_{i-2}$ for $i \geq 3$ with the boundary conditions $f_1 = f_2 = 1$.

RNA (RiboNucleic Acid): Single stranded molecule serving as intermediary between DNA and protein. Some RNAs also possess catalytic activity and complex with proteins to form *ribozymes*, the most important of which is the ribosome, the site of protein synthesis. By convention a given RNA sequence, e.g. `AGU`, is written in the 5' to 3' direction.

rooted tree: Refers to a phylogeny with known position of the last common ancestor.

selective sweep: The evolutionary process during which an advantageous allele gets fixed in a population.

sensitivity: In the context of gene prediction or database searching this refers to the fraction of true targets returned.

shotgun sequencing: Method for sequencing long sequences by analyzing a large number of small random fragments and automatically assembling these.

SNP (Single Nucleotide Polymorphism): Detected by analyzing homologous positions in two or more DNA sequences. If a position is occupied by more than one nucleotide a SNP has been found.

specificity: In the context of gene prediction or database searching this refers to the fraction of true predictions or database hits.

splicing: Removal of introns from primary transcripts. In eucaryotes this takes place in the nucleus and the spliced RNAs are exported to the cytoplasm where they are translated.

STS (Sequence Tagged Site): A molecular marker consisting of a stretch of perhaps 100 nucleotides of known sequence that is unique within the genome. Often used for genome mapping.

substitution: Mutation that leads to an observable allele.

suffix tree: Data structure for indexing a text such that it can be searched for a pattern in time proportional to the length of the pattern irrespective of the length of the text.

SwissProt: Reference database of protein sequences maintained by the Swiss Institute of Bioinformatics and the European Bioinformatics Institute (EBI).

synonymous substitution: Substitution in a coding DNA sequence that leaves the encoded protein sequence unchanged.

taxon: A group of organisms, a taxonomic unit.

telomere: Terminal regions of chromosomes for protecting the ends of linear chromosomes consisting of a characteristic tandem repeat, for example $(\texttt{TTAGGG})_n$.

text: In the context of string searching this refers to the string in which a pattern is to be found.

traceback: In the context of optimal alignment, this refers to the process of extracting the alignment from the filled in alignment matrix by following the back pointers specified during the forward phase of the algorithm.

transition: Mutation among members of the same chemical class of nucleotides, i.e. among purines (`A`, `G`) or among pyrimidines (`C`, `T`).

transversion: Mutation between members of the chemical classes of nucleotide bases, i.e. between purines (`A`, `G`) and pyrimidines (`C`, `T`).

ultrametric distances: Pairwise distances between taxa evolving at the same rate.

unrooted tree: Phylogeny with unknown position of the last common ancestor.

UPGMA (Unweighted Pair-Group Method using an arithmetic Average): Method for reconstructing a phylogeny from ultrametric pairwise distance data.

weight matrix: Matrix of the frequency (or some function thereof) of each possible residue along the aligned members of a protein family.

Author Index

Aguadé, M. 245, 257
Aho, A. 49
Altschul, S. F. 14, 82
Avery, O. T. 117

Baltimore, D. 70
Barker, W. C. 65
Bianchi, N. O. 179
Brenner, S. 11
Burge, C. 139
Burset, M. 121

Cantor, C. R. 151, 152, 160, 161, 174–176, 178, 180, 181, 186
Chakravarti, A. 169
Chomsky, N. 11
Clark, A. G. 215
Corasick, M. 49
Crick, F. H. C. 63, 101, 117
Crow, J. F. 206, 207, 211

Darwin, C. 191, 233
Dayhoff, M. O. 18, 65
Doolittle, R. F. 70, 97
Durbin, R. 114

Efron, B. 165
Ewens, W. J. 209–211, 215, 245, 275

Feller, W. 275
Felsenstein, J. 160, 165, 167
Feng, D.-F. 97
Fisher, R. A. 160, 187, 189, 191, 194, 202, 213, 253
Fitch, W. M. 157
Fu, Y.-X. 251, 252, 257
Fuerst, P. A. 207

Gilbert, W. 16
Gillespie, J. H. 187
Grant, G. R. 275
Griffith, F. 117
Griffith, R. C. 235
Gusfield, D. 62

Haldane, J. B. S. 192, 202
Hardy, G. H. 196, 197, 216
Hartl, D. L. 215
Hein, J. 242
Holmes, E. C. 167
Hudson, R. R. 235, 242, 245, 257

Jukes, T. H. 117, 151, 152, 160, 161, 174–176, 178, 180, 181, 186, 192

Kaplan, N. L. 235
Karlin, S. 82
Kendrew, J. C. 91
Kimura, M. 172, 174, 180, 186, 192, 200, 206, 207, 211
King, J. L. 117, 143, 192
Kingman, J. F. C. 217
Knuth, D. E. 46
Kreitman, M. 137, 189, 245, 253, 257

Lewontin, R. C. 245
Li, W. 80, 140, 251, 252, 257
Li, W.-H. 185
Lipman, D. J. 14

Majoram, P. 235
Maynard Smith, J. 7
McDonald, J. H. 253, 257
Mendel, G. 117
Moloney, G. 65

Needleman, S. B. 14
Nordborg, M. 242

Ohta, T. 172, 200

Page, R. D. M. 167
Pamilo, P. 179
Pauling, L. 172
Pearson, W. R. 14
Perutz, M. 91
Provine, W. B. 215

Rabiner, L. R. 107, 109, 114
Rous, P. 69

Sanger, F. 16
Shierup, M. 242
Swofford, D. L. 167

Tajima, F. 248, 249, 251, 253, 257
Temin, H. 70

Vogt, P. 70

Waterman, M. S. 14, 40
Watson, J. D. 63, 117
Watterson, G. A. 211, 229, 232, 233
Weinberg, W. 196, 197, 216
Wiuf, C. 242
Wright, S. 187, 189, 191, 194, 202, 215
Wunsch, C. D. 14

Zhang, M. Q. 139
Zuckerkandl, E. 172

Subject Index

- acceptor site 124
- adaptation 214
- additive distances 151, 155, 166
- adenine 281
- *Adh* locus 263
  - donor site prediction 130
  - GenBank entry 119
  - gene structure 120
  - genetic diversity 190, 247, 254
- affine gap model 36
- Aho-Corasick-algorithm 49
- alignment 11, 14, 39, 44, 118, 184
  - gaps 170
  - global 12, 30, 37, 39, 67, 68, 262
  - global/local 72
  - heuristic 13, 67, 69, 79, 86
  - local 12, 27, 35, 39, 69, 82, 86
  - multiple *see* also separate entry, 91
  - number of alignments 27, 261
  - optimal 13, 65, 66, 79, 262
  - overlap 12, 33, 39
  - pairwise 91, 137, 259
  - progressive 97, 113
  - score 15, 21, 25, 30, 38, 83, 86, 262
  - spliced 135
  - types of 12
  - uniqueness 33
  - variable positions 158
- allele 187
  - fixation of 195
  - frequency 199, 202
  - loss of 195, 200
  - singleton 211
- α helix 91, 92
- amino acids 16, 281
  - background frequencies 20, 22, 82
  - nomenclature 282
  - physico-chemical properties 283
- *Amoeba dubia* 56
- analogy 11, 72
- *Arabidopsis thaliana* 35, 256
- archebacteria 1
- array 74
- assembly 34, 57
- average linkage clustering 152

- back pointer 33
- bacteria 1
- bacteriophage
  - ϕX174 35, 263
  - λ 35
- balancing selection 249
- Bayes' Formula 276
- binary search 55
- `bioinformer` 4, 259
- BLAST 76, 79, 86, 135, 285
  - bit score 85
  - E-value 83
  - maximal segment pair 76, 79
- `blastclust` 80
- BLOCKS database 22
- BLOSUM matrix 26, 89
- bootstrap 164, 166, 167, 269
- boundary condition 27
- branch and bound 163

- C-value paradox 56

Caenorhabditis elegans 35
central dogma 1, 3
`clustalw` 91, 105, 113, 285
coalescence 222
coalescent 222, 242, 251
  theory 222, 273
  tree depth 225, 226
  vs. phylogeny 224
  with recombination 235, 236
  with selection 231
  with selection and recombination 238
codon 6, 16
  bias 126, 127, 134
  space 17
computational biology 1
conditional expectation 277
coverage 34
cytosine 281

deterministic selection equation 213
diffusion process 199
`dists1` 180
`DIVERGE` 179
divergence 188, 245, 247, 248
  time 172
diversity 245, 247, 248
DNA structure 280
`DnaSP` 257, 285
domains of life 2
dominance 214
donor site 124, 128
dotplot 73, 266
drift *see* genetic drift
Drosophila melanogaster 12, 35, 137, 247
  Adh locus 135, 189
  genetic diversity 191
Drosophila simulans 137
dynamic programming 14, 30, 40, 75, 83, 86, 94, 110, 131, 160

effective number of alleles 207
EMBOSS 104, 106, 285
Escherichia coli 68, 86, 256
`est_genome` 134
estimator 277
eucaryotes 1
evolution 3, 217, 268
  adaptive 191
  non-adaptive 191
  of microbes 193
  rate of 205
evolutionary relatedness 184
evolvability 7
Ewens' sampling formula 209
exon 117, 121, 122, 135
  chaining 131, 132
  reading frame 125
  remainder 125
  types 126
expectation 277
extreme value distribution 81

FASTA 73, 79, 265, 269, 285
finite alleles model 202, 215
Fitch-parsimony 157
fixation 204, 215, 239, 249
  probability of 197, 214
  rate of 203, 205
  time to 197–199, 203, 214
fixed difference 253
forkhead box protein 2 182, 184
four point criterion 151
FOXP2 *see* forkhead box protein 2
fruit fly *see Drosophila melanogaster*

G/C-rich region 109, 267
gap 12, 40
  affine model 15, 36
  linear model 15
GenBank database 66, 119, 120, 126, 171
gene 117, 139
  duplication 80, 138
  families 79
  genealogy 223, 252
  knockout 79
  models 121
  pool 194, 218
  singleton 80
gene prediction 117, 138, 139
  ab initio 118, 124, 139
  accuracy 123
  algorithms 120
  chimeric 122
  comparative 118, 135, 139
  joined gene 123
  sensitivity 130
  sensitivity and specificity 121, 139
  specificity 130

  - splice sites 126
  - split gene 122, 137, 138
- genealogy 188, 217, 242
- generalized suffix tree 59, 68, 86
- `GeneSeqer` 135
- genetic code 284
- genetic diversity 187–189, 191, 193, 200, 207, 215, 239
- genetic drift 188, 192, 194, 196, 201, 214, 215, 270
- genetic hitchhiking 239, 249
- genetic load 192
- `genewise` 134
- genome 2, 35, 56
- `GenScan` 134, 285
- `gff2aplot` 285
- globins 91, 101, 180
  - alignment 92, 96
  - δ *vs.* γ 74
  - phylogeny 102, 143, 144
  - profile 106
- guanine 281
- guide tree 98

- *Haemophilus influenzae* 35
- Hamming distance 203
- Hardy-Weinberg law 197
- hash table 74, 265
- heterozygosity 193, 200, 207, 208, 230, 232
  - decay 201
  - per bp 240
- heterozygotes 202
- hidden Markov model 101, 106, 108, 114, 134, 267
  - and sequence alignment 113
  - detection problem 110
  - elements 108
  - forward procedure 109
  - scoring problem 109
  - searching for homologous sequences 112
  - three fundamental problems 109
- hitchhiking mapping 242
- HKA test *see* Hudson-Kreitman-Aguadé test
- HMM *see* hidden Markov model
- homeodomain 12
- homogeneity 195
- homology 11, 35, 44, 72, 91, 92, 106, 111, 114, 169, 271
- homoplasy 151, 158, 166
- homozygosity 206
- homozygotes 202
- horizontal gene transfer 69
- *Hox* genes 12, 28
- human genome 35, 43, 57
- human immuno-deficiency virus (HIV) 70

- indels 12, 14, 39, 62, 71, 187, 208
- index of association 255, 257
- infinite alleles model 203, 208, 209, 215
- infinite sites model 203, 215, 224, 229, 254
- information content 129
- interspersed repeats 57
- intron 117
  - phase 125

- Jukes-Cantor
  - distance 152, 269
  - model 151, 160, 174, 176, 178

- keyword tree 48, 49, 62, 71, 76

- language 182
- `LIAN` 257
- likelihood 159
  - -mapping 149, 166
  - ratio test 161, 167
- linkage 255
  - disequilibrium 255, 257
  - equilibrium 246, 255
- log-odds score 22, 82, 85
- longest common substring 60

- Markov chain 107, 114, 197, 198
- match 44, 259
- maximal repeats 59, 68
- maximal unique matches 68
- mean fitness 213, 214
- microsatellite 56, 189
- midpoint rooting 144, 145
- minisatellite 189
- mismatch 44, 98, 259
- mismatch distribution 231
- mismatch percentage 172
- molecular clock 143, 144, 166, 172
- molecular evolution 171

moment 277
most recent common ancestor 221, 222, 226
`ms` 233, 234, 238, 285
`msa` 96
multiple sequence alignment 91, 98, 101, 107, 111, 114, 145
  global 98
  heuristic 93, 97
  optimal 94
  progressive 97
`MUMmer` 67, 72, 79, 285
Mus musculus 35
mutation 7, 14, 171
  advantageous 213
  amino acid 19
  frequency spectrum 229
  models of 201
  nucleotide 16
  rate of 143, 171, 225
  singleton 230, 237, 252, 274
mutation drift equilibrium 192, 203, 215
mutational distance 16, 146
myoglobin 91, 93

natural selection *see* selection
nearest neighbor interchange 162
neighbor-joining algorithm 155, 156, 166
Neisseria gonorrhoeae 256
neutral molecular evolution 192, 201, 207, 215
neutral mutation hypothesis 191, 210, 245, 248
  Ewens-Watterson test 211
  Fu and Li's test 251
  Hudson-Kreitman-Aguadé test 245
  McDonald-Kreitman test 253
  Tajima's test 248
neutral mutation parameter 246
neutral theory *see* neutral molecular evolution; neutral mutation hypothesis
nucleic acids 279
nucleoside 279
nucleotide 279
  ambiguity code 281
null distribution 164
number of alleles 205, 210, 211
number of substitutions 181

orthologous sequences 135, 137
orthology 13, 169, 180
outgroup 144, 166

palindrome 73
PAM matrix 21
Pan troglodytes 35
panmixis 194, 197, 217
paralogous sequences 135
paralogy 169
parameter 277
parsimony 157, 163
PCR primers 92
peptide biosynthesis 283
percent accepted mutations 18
PFAM database 113
phage *see* bacteriophage
PHYLIP 285
phylogeny 93, 143, 268
  bifurcating 143, 166
  counting trees 148
  distance methods 150, 166
  length of 157, 159, 166
  maximum likelihood 159, 161, 166
  maximum parsimony 157, 158, 166
  quartet analysis 148
  rate equality 144
  reconstruction 183
  rooted 144, 145, 149, 150, 166
  topologies 147
  tree space 161
  unrooted 145, 149
  vs. coalescent 224
platelet-derived growth factor 70
point mutation 14, 39
Poisson distribution 83
polymorphism 183, 188, 191, 237, 253
population 187, 271
  effective size 208
  genealogy 221
  genetics 202
position weight matrix 126, 129
prefix 30
primitive donor finder 126, 129
probability 275–276
probability distribution 245, 277
`procrustes` 134
profile 101, 105, 113
  alignment 105

  analysis 101
  hidden Markov model 106, 111, 114
PROSITE database 44
protein
  3D-structure 93
  biosynthesis 283
  domain 99
  families 80, 92, 98, 101, 111, 112
  primary structure 101
  sequence 16
proteome 80
pruning and regrafting 163
`psi-blast` 106
purine 281
pyrimidine 281

query sequence 74

random event 275
random genetic drift *see* genetic drift
raw score 85, 87
recombination 233, 245, 253, 254
  minimum number of events 254, 257
regular expression 44
resampling
  with replacement 193, 271
  without replacement 256
reverse transcriptase 70
ribose 280

Saccharomyces cerevisiae 35, 56
sarcoma virus 65, 69
scaled mutation rate 207, 273
score matrices *see* substitution matrices
scoring scheme 18, 30, 40, 75, 94
segregating sites 224, 225, 229, 233, 238
  number of 248, 273
selection 3, 191, 212, 237, 270
  background 237
  balancing 231, 238
  cost of 192
  directional 231, 237
  diversifying 179
  fundamental theorem 214
  negative 237
  positive 214, 237
  purifying 179, 237, 249
  weak 214
selective sweep 249, 251
sensitivity 77, 121, 130, 139
sequence 171
  annotation 118
  logo 129, 130
  motifs 98
  repetitive 56
  space 3–6
  variation 169, 184
`SGP-2` 285
shotgun sequencing 34, 35
sickle-cell anemia 237
signature sequences 44
`sim4` 134
single nucleotide polymorphisms 187, 224, 250, 252, 254, 273
  bi-allelic 188
  density of 237
`Slam` 137, 285
SNP *see* single nucleotide polymorphisms
specificity 77, 121, 130, 139
src 70
star phylogeny 146, 237, 239
state transition matrix
  amino acids 19
  nucleotides 174
statistical geometry 145, 146
Streptococcus pneumoniae 117
string 30
string matching 45, 55, 62, 263
  exact 43, 44, 86, 265
  inexact 43, 44, 61, 62
  k-error 71, 79
  k-mismatch 61, 71
  naïve 44, 263
  set 49
  with keyword tree 50
  with suffix tree 51, 263
subject sequence 75
subsequence 31, 68, 86
substitution 171, 189, 192
  in *FOXP2* 183
  non-synonymous 19, 178, 253
  per site 176
  rate of 171, 175, 176
  ratio non-synonymous to synonymous 182, 185, 253
  synonymous 178, 180, 192, 253
substitution matrices 16, 18, 21, 23, 75, 82, 84, 102, 259

  application of 27, 28
  BLOSUM 22, 25, 40, 76, 82, 85, 260
  for profile analysis 104
  log-odds score 22
  PAM 18, 21, 40, 76, 85, 260
  PAM *vs.* BLOSUM 25, 26
substring 30, 35
  repeated 57
  unique 58
sufficient statistic 211
suffix 30
suffix array 55
suffix tree 51, 59, 62, 264
  construction 54
  generalized 59
  left-divergent node 59
  lowest common ancestor 60, 61
  string depth 58, 264
  string matching with 51
sum-of-pairs score 94
SwissProt database 106

Tajima's D 251
target frequencies 85
three point criterion 151
thymine 281
traceback 30, 33, 36
transcript variants 139
transcription signals 124
translation signals 124
transposition 69
tree 46, 47, 55, 222
tree length 235
type III secretion pathway 68

ultrametric distances 151, 154, 166
universal ancestor 217
UPGMA algorithm 144, 153, 155, 269
uracil 281

variance 277
Vibrio 67
Viterbi algorithm 110, 111, 113

wild type 3
wild-type allele 187, 213
Wright-Fisher model 194, 214, 217, 218, 242, 271
  absorbing states 195
  absorption time 198, 199
  lineages 218

yeast 80

Z-algorithm 45, 263