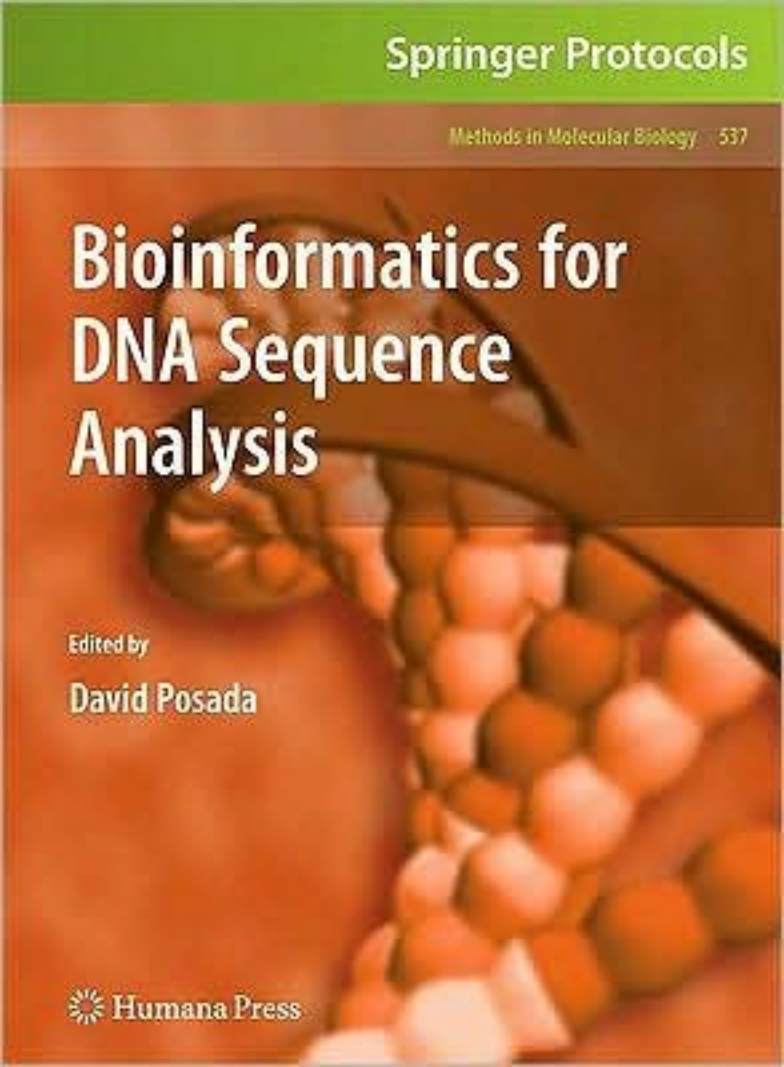
Springer Protocols

Methods in Molecular Biology 537

Bioinformatics for DNA Sequence Analysis



Edited by

David Posada

Humana Press

METHODS IN MOLECULAR BIOLOGY™

Series Editor
John M. Walker
School of Life Sciences
University of Hertfordshire
Hatfield, Hertfordshire, AL10 9AB, UK

For other titles published in this series, go to
www.springer.com/series/7651

METHODS IN MOLECULAR BIOLOGY™

Bioinformatics for DNA Sequence Analysis

Edited by

David Posada

Departamento de Genética, Bioquímica e Inmunología,
Facultad de Biología, Universidad de Vigo,
Vigo, Spain

Humana Press

Editor
David Posada
Departamento de Genética
Bioquímica e Inmunología
Facultad de Biología
Universidad de Vigo
Vigo
Spain
dposada@uvigo.es

ISSN 1064-3745 e-ISSN 1940-6029
ISBN 978-1-58829-910-9 e-ISBN 978-1-59745-251-9
DOI 10.1007/978-1-59745-251-9

Library of Congress Control Number: 2008941278

Printed on acid-free paper

springer.com

To Mónica and Lucas

Preface

The recent accumulation of information from genomes, including their sequences, has resulted not only in new attempts to answer old questions and solve longstanding issues in biology, but also in the formulation of novel hypotheses that arise precisely from this wealth of data. The storage, processing, description, transmission, connection, and analysis of these data has prompted bioinformatics to become one the most relevant applied sciences for this new century, walking hand-in-hand with modern molecular biology and clearly impacting areas like biotechnology and biomedicine.

Bioinformatics skills have now become essential for many scientists working with DNA sequences. With this idea in mind, this book aims to provide practical guidance and troubleshooting advice for the computational analysis of DNA sequences, covering a range of issues and methods that unveil the multitude of applications and relevance that Bioinformatics has today. The analysis of protein sequences has been purposely excluded to gain focus. Individual book chapters are oriented toward the description of the use of specific bioinformatics tools, accompanied by practical examples, a discussion on the interpretation of results, and specific comments on strengths and limitations of the methods and tools. In a sense, chapters could be seen as enriched task-oriented manuals that will direct the reader in completing specific bioinformatics analyses.

The target audience for this book is biochemists, and molecular and evolutionary biologists that want to learn how to analyze DNA sequences in a simple but meaningful fashion. Readers do not need a special background in statistics, mathematics, or computer science, just a basic knowledge of molecular biology and genetics. All the tools described in the book are free and all of them can be downloaded or accessed through the web. Most chapters could be used for practical advanced undergraduate or graduate-level courses in bioinformatics and molecular evolution.

The book could not start in another place than describing one of the most widespread bioinformatics tool: BLAST (Basic Local Alignment Search Tool). Indeed, one of the first steps in the analysis of DNA sequences is their collection. Therefore, **Chapter 1** guides the reader through the recognition of similar sequences using BLAST. Next, the use of OrthologID for understanding the nature of this similarity is described in **Chapter 2**, followed by a **Chapter 3** about one of the most important stages in most bioinformatics pipelines, the alignment, which shows the basis and the application of the program MAFFT. The next set of chapters is intimately related to the study of molecular evolution. Indeed, the DNA sequences that we see today are the result of this process. In **Chapter 4**, SeqVis is used to detect compositional changes in DNA sequences through time, while **Chapter 5** is focused on the selection of models of nucleotide substitution using jModelTest. Precisely the use of these models for phylogenetic reconstruction is described in **Chapter 6**, which capitalizes upon the estimation of maximum likelihood phylogenetic trees with Phyml. Indeed, the estimation of phylogenies is often the first step in many evolutionary analyses. How to combine multiple trees in a single supertree is the basis of **Chapter 7**, which explains

the use of the program Clann. Next, **Chapters 8 and 9** are centered on the characterization of two key evolutionary processes acting on DNA sequences. The use of the server Datamonkey for the detection of selection is described in **Chapter 8**, while **Chapter 9** shows the nuts and bolts of the detection of recombination using RDP3.

The study of codon usage, which has provided many important insights at the genomic scale, is deciphered in **Chapter 10** using CodonExplorer, an interactive data base, while **Chapter 11** explains how differences in the genetic code can be detected using GenDecoder. The next chapters are related to the annotation of genomes, an essential requisite for many other analyses. In **Chapter 12**, we learn how to predict genes using GeneID, while in **Chapter 13** the identification of regulatory motifs with A-Glam is described. **Chapter 14** then explains the use of the UCSC genome browser and its applications, for example, to characterize a gene or to explore conserved elements. The discovery of single nucleotide polymorphisms (SNPs) and simple sequence repeats (SSRs) with bioinformatics tools SNPServer, dbSNP, and SSR Taxonomy Tree is the subject of **Chapter 15**, and **Chapter 16** highlights the use of Censor and RepeatMasker for the detection and characterization of transposable of sequences in eukaryotic genomes. To end the book, **Chapter 17** explains how to make the most of DnaSP for the analysis of DNA sequences in populations.

I am very grateful to all the authors, the fundamental piece, who have put a lot of effort replying patiently to all my queries. Hopefully, the result has been a set of clear and useful chapters that will be of help to other scientists. I want to thank all of them for sharing their time, wisdom and expertise. Finally, I want to thank John Walker, the editor of the series, for his continuous advice.

Vigo, July 2008 ***David Posada***

Contents

Contributors

Federico Abascal • *Departamento de Genética, Bioquímica e Inmunología, Facultad de Biología, Universidad de Vigo, Vigo, Spain*

Josep F. Abril • *Departament de Genètica, Facultat de Biologia, Universitat de Barcelona, Spain*

George Asimenos • *Department of Computer Science, Stanford University, Stanford, CA, USA*

Jacqueline Batley • *Australian Centre for Plant Functional Genomics, School of Land, Crop and Food Sciences, University of Queensland, Brisbane, Australia*

Enrique Blanco • *Departament de Genètica, Facultat de Biologia, Universitat de Barcelona, Spain*

Joanna C. Chiu • *Department of Molecular Biology and Biochemistry, Rutgers University, Piscataway, NJ, USA*

Mark Clement • *Department of Computer Science, Brigham Young University, Provo, UT, USA*

Gloria Coruzzi • *Department of Biology, New York University, New York, NY, USA*

Keith A. Crandall • *Department of Biology, Brigham Young University, Provo, UT, USA*

Christopher J. Creevey • *EMBL Heidelberg, Heidelberg, Germany*

Frederic Delsuc • *Institut des Sciences de l'Evolution de Montpellier (ISEM), UMR 5554-CNRS, Université Montpellier I I, Montpellier, France*

Rob Desalle • *Sackler Institute of Comparative Genomics, American Museum of Natural History New York, NY, USA*

Jean-François Dufayard • *Laboratoire d'Informatique, de Robotique et de Microélectronique de Montpellier (LIRMM). UMR 5506-CNRS, Université Montpellier I I, Montpellier, France*

David Edwards • *Australian Centre for Plant Functional Genomics, School of Land, Crop and Food Sciences, University of Queensland, Brisbane, Australia*

Mary Egan • *Department of Biology, Montclair State University, Montclair, NJ, USA*

Simon D.W. Frost • *Antiviral Research Center, Department of Pathology, University of California San Diego, La Jolla, CA, USA*

Olivier Gascuel • *Laboratoire d'Informatique, de Robotique et de Microélectronique de Montpellier (LIRMM). UMR 5506-CNRS, Université Montpellier I I, Montpellier, France*

Stéphane Guindon • *Laboratoire d'Informatique, de Robotique et de Microélectronique de Montpellier (LIRMM). UMR 5506-CNRS, Université Montpellier II, Montpellier, France; Department of Statistics, University of Auckland. Auckland, New Zealand*

Micah Hamady • *Department of Computer Science, University of Colorado, Boulder, CO, USA*

Joshua Wing Kei Ho • *School of Information Technologies, University of Sydney, Sydney, Australia; NICTA, Australian Technology Park, Eveleigh, Australia*
Ahsan Huda • *School of Biology, Georgia Institute of Technology, Atlanta, GA, USA*
Vivek Jayaswal • *Centre for Mathematical Biology, Sydney Bioinformatics, School of Mathematics and Statistics, University of Sydney, Sydney, Australia*
Lars Somer Jermiin • *School of Biological Sciences, Centre for Mathematical Biology and Sydney Bioinformatics, University of Sydney, Sydney, Australia*
I. King Jordan • *School of Biology, Georgia Institute of Technology, Atlanta, GA, USA*
Kazutaka Katoh • *Digital Medicine Initiative, Kyushu University, Fukuoka 812-8582, Japan*
Rob Knight • *Department of Chemistry and Biochemistry, University of Colorado, Boulder, CO, USA*
David Landsman • *Computational Biology Branch, National Center for Biotechnology Information, National Library of Medicine, National Institutes of Health, Bethesda, MD, USA*
Kwok Wai Lau • *School of Biological Sciences, University of Sydney, Sydney, Australia*
Ernest K. Lee • *Department of Biology , New York University, New York, NY, USA*
Leonardo Mariño-Ramírez • *Computational Biology Branch, National Center for Biotechnology Information, National Library of Medicine, National Institutes of Health, Bethesda, MD, USA*
Darren P. Martin • *Institute of Infectious Disease and Molecular Medicine, University of Cape Town, Observatory, Cape Town, South Africa*
James O. Mcinerney • *Department of Biology, National University of Ireland Maynooth, Co. Kildare, Ireland*
Kit J. Menlove • *Department of Biology, Brigham Young University, Provo, UT, USA*
Jonathan Pevsner • *Department of Neurology, Kennedy Krieger Institute, Baltimore, MD, USA*
Sergei L. Kosakovsky Pond • *Antiviral Research Center, Department of Pathology, University of California San Diego, La Jolla, CA, USA*
Art F.Y. Poon • *Antiviral Research Center, Department of Pathology, University of California San Diego, La Jolla, CA, USA*
David Posada • *Departamento de Genética, Bioquímica e Inmunología, Facultad de Biología, Universidad de Vigo, Vigo, Spain*
Julio Rozas • *Departament de Genètica, Facultat de Biologia, Universitat de Barcelona, Barcelona, Spain*
John L. Spouge • *Computational Biology Branch, National Center for Biotechnology Information, National Library of Medicine, National Institutes of Health, Bethesda, MD, USA*
Noboru Sueoka • *Department of Molecular, Cellular, and Developmental Biology, University of Colorado, Boulder, CO, USA*
Kannan Tharakaraman • *Computational Biology Branch, National Center for Biotechnology Information, National Library of Medicine, National Institutes of Health, Bethesda, MD, USA*
Hiroyuki Toh • *Medical Institute of Bioregulation, Kyushu University, Fukuoka, Japan*

Jesse Zaneveld • *Department of Molecular, Cellular, and Developmental Biology, University of Colorado, Boulder, CO, USA*

Rafael Zardoya • *Departamento de Biodiversidad y Biología Evolutiva, Museo Nacional de Ciencias Naturales, Madrid, Spain*

Chapter 1

Similarity Searching Using BLAST

Kit J. Menlove, Mark Clement, and Keith A. Crandall

Abstract

Similarity searches are an essential component of most bioinformatic applications. They form the bases of structural motif identification, gene identification, and insights into functional associations. With the rapid increase in the available genetic data through a wide variety of databases, similarity searches are an essential tool for accessing these data in an informative and productive way. In this chapter, we provide an overview of similarity searching approaches, related databases, and parameter options to achieve the best results for a variety of applications. We then provide a worked example and some notes for consideration.

Key words: BLAST, sequence alignment, similarity search.

1. Introduction

1.1. An Introduction to Nucleotide Databases

Perhaps the central goal of genetics is to articulate the associations of phenotypes of interest with their underlying genetic components and then to understand the relationship between genetic variation and variation in the phenotype. This goal has been buoyed by the tremendous increase in our ability to obtain molecular genetic data, across both populations and species. As methods of gathering information about various aspects of biological macromolecules arose, biological information became abundant and the need to consolidate and make this information accessible became increasingly apparent. In the early 1960s, Margaret Dayhoff and colleagues at the National Biomedical Research Foundation (NBRF) began collecting information on protein sequences and structure into a volume entitled *Atlas of Protein Sequence and Structure (1)*. Since that beginning, databases have been an important and essential part of biological and biochemical research.

David Posada (ed.), *Bioinformatics for DNA Sequence Analysis,* Methods in Molecular Biology 537

DOI 10.1007/978-1-59745-251-9_1

By 1972, the size of the Atlas had become unwieldy, so Dr. Dayhoff, a pioneer of bioinformatics, developed a database infrastructure into which this information could be funneled. Though nucleotide information was included in the Atlas as early as 1966 *(2)*, its bulk was comprised of amino acid sequences with structural annotation.

1.2. International Nucleotide Sequence Database Collaboration: DDBJ, EMBL, and GenBank

It was not until 1982 that databases were developed with the express purpose of storing nucleotide sequences by the European Molecular Biology Laboratory (EMBL: http://www.embl.org/) in Europe and the National Institutes of Health (NIH – NCBI: http://www.ncbi.nlm.nih.gov/) in North America. Japan followed suit with the creation of their DNA Databank (DDBJ: http://www.ddbj.nig.ac.jp/) in 1986. A sizeable amount of sharing naturally occurred between these three databases and the Genome Sequence Database, also in North America, a condition that led to their coalition in 1988 under the title International Nucleotide Sequence Database Collaboration (INSDC). They still remain very distinct entities, but in the 1988 meeting, they established policies to govern the formatting of and stewardship over the sequences each receives. Their current policies include unrestricted access and use of all data records, proper citation of data originators, and the responsibilities of submitters to verify the validity of the data and their right to submit it. The INSDC currently contains approximately 80 billion base pairs (bp) (not including whole-genome shotgun sequences) and nearly 80 million sequence entries. Including shotgun sequences (HTGS), it passed the 100-gigabase mark on August 22, 2005, and contains approximately 200 billion bp as of September 2007. For more than 10 years, the amount of data in these databases doubled approximately every 18 months. This expansion has begun to level off as our capacity for high-throughput sequencing is gradually reaching a maximum. The next redoubling of the data is expected to occur in approximately 4 years (**Fig. 1.1**).

1.3. Other Nucleotide Sequence Databases

Since the first nucleotide databases were initiated by EMBL and NIH (now held by NCBI), many DNA databases have been formed to cater to the needs of specialized research groups. The 2007 Database issue of *Nucleic Acid Research* contained 109 nucleotide sequence databases that met the standards required to be included in its listing *(3)*. These databases are typically developed to include ancillary data associated with the genetic data, such as patient or specimen information, including clinical information, images, downstream analyses. Many do not meet the standards of "quality, quantity and originality of data as well as the quality of the web interface" that are required to be considered for the issue *(4)*. Even more are privately held to permit access of costly data to a select few. All in all, the number of DNA databases is astounding and steadily increasing as we find new, powerful ways to gather, store, and utilize the pieces that comprise the puzzle of life.

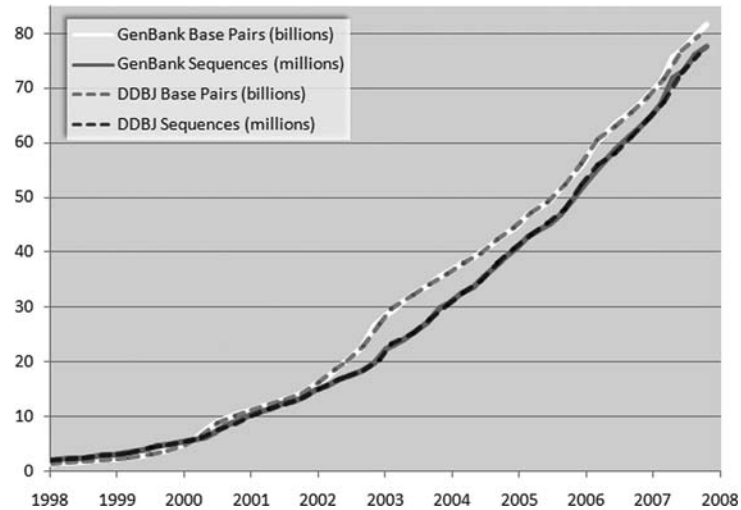


Fig. 1.1. Growth of GenBank and DDBJ genetic databases over the past 10 years. The INSDC databases have grown, over the past 10 years, approximately 168-fold in total number of base pairs. While in the past the number of entries in INSDC databases doubled approximately every 2 years, a simple second-order polynomial regression (R^2=0.9995) of the data over the past 10 years indicates that the next redoubling will take a little over 4 years. This graph does not include HTG data.

2. Program Usage

2.1. Database File Formats

One of the largest sources of diversity among DNA databases lies in their file formats. While great efforts have been made to standardize file formats, the various types and purposes of sequence information and annotation entreat customized file types.

2.1.1. FASTA Format

First used with Pearson and Lipman's FASTA program for sequence comparison *(5)*, the FASTA file format is the simplest of the widely used formats available through the INSDC. It is composed of a definition or description line followed by the sequence. The definition line begins with a greater-than symbol (>) and marks the beginning of each new entry. The information following the greater-than symbol varies according to its source. Generally, an identifier follows (**Table 1.1**), after which optional description words may be included. If the sequence is retrieved through NCBI's databases, a GI number precedes the identifier. Though it is recommended that the definition line be no greater than 80 characters, various types and levels of information are often included. The definition line is followed by the DNA sequence itself, in single or multi-line format. Nucleotides are represented by their standard IUB/IUPAC codes, including ambiguity codes (**Table 1.2**).

Table 1.1
FASTA File sequence identifiers. Information from the NCBI Handbook *(25)*

Database name	Identifier syntax
GenBank	gb|*accession.version*
EMBL	emb|*accession.version*
DDBJ	dbj|*accession.version*
NCBI RefSeq	ref|*accession.version*
PDB	pdb|*entry*|*chain*
Patents	pat|*country*|*number*
NBRF PIR	pir||*entry*
SWISS-PROT	sp|*accession*|*entry*
Protein Research Foundation	prf|*name*
GenInfo Backbone Id	bbs|*number*
General database identifier	gnl|*database*|*identifier*
Local Sequence identifier	lcl|*identifier*

Table 1.2
IUB/IUPAC nucleotide and ambiguity codes

A	**a**denosine	M	A or C (**am**ino)	V	A, C, or G
C	**c**ytidine	K	G or T (**k**eto)	H	A, C, or T
G	**g**uanine	R	A or G (pu**r**ine)	D	A, G, or T
T	**t**hymidine	Y	C or T (p**y**rimidine)	B	C, G, or T
U	**u**ridine	S	A or T (**s**trong)	–	Gap of indeterminate length
		W	C or G (**w**eak)	N	A, C, G, or T (a**n**y or unk**n**own)

2.1.2. Flat File Format

GenBank, EMBL, and DDBJ each have their own flat file format, but contain basically the same information. They are all based upon the Feature Table, which can be found at http://www.ncbi.nlm.nih.gov/collab/FT. For references to these file types, see *(6–9)*.

2.1.3. Accession Numbers, Version Numbers, Locus Names, Database Identifiers, etc.

The standard for identifying a nucleotide sequence record is by an *accession.version* system where the *accession number* is an identifier of two letters followed by six digits and the *version* is an incremental number indicating the number of changes that have been

made to the sequence since it was first submitted. Locus names (*see* **Note 1**) are older, less standardized identifiers whose original purpose was to group entries with similar sequences *(10)*. The original locus format was intended to hold information about the organism and other common group characteristics (such as gene product). That ten-character format is no longer able to hold such information for the large number and variety of sequences now available, so the locus has become yet another unique identifier often set to be the same value as the accession number. Database identifiers are simply two- or three-character strings that serve to indicate which database originally received and stored the information. The database identifier is the first value listed in the FASTA identifier syntax (**Table 1.1**).

When a sequence is first submitted to GenBank, it is submitted with several defined features associated with the sequence. Some include CDS (coding sequence), RBS (ribosome binding site), rep_origin (origin of replication), and tRNA (mature transfer RNA) information. A translation of protein coding nucleotide sequences into amino acids is provided as part of the features section. Likewise, labeling of different open reading frames, introns, etc., are all part of the table of features. A list of features and their descriptions, formats, and conventions that were agreed upon by INSDC can be found in the Feature Table (*see* **Section 2.1.2**).

2.2. Smith–Waterman and Dynamic Programming

In 1970, Needleman and Wunsch adapted the idea of dynamic programming to the difficult problem of global sequence alignment *(11)*. In 1981, Smith and Waterman adapted this algorithm to local alignments *(12)*. A global alignment attempts to align two sequences throughout their entire length, whereas a local alignment aligns regions of two sequences where high similarity is observed. Both methods involve initializing, scoring, and tracing a matrix where the rows and columns correspond to the bases or residues of the two sequences being aligned (**Fig. 1.2**). In the local alignment case, the first row and the first column are filled with zeroes. The remaining cells are filled with a metric value recursively derived from neighboring values:

$$\max\begin{cases} 0 \\ \text{left neighbor} + \text{gap penalty} \\ \text{top neighbor} + \text{gap penatly} \\ \text{top-left neighbor} + \text{match/mismatch score} \end{cases}$$

If the current cell corresponds to a match (identical bases), the match score is added to the value from the diagonal neighbor, otherwise the mismatch score is used. The gap penalty and mismatch scores are generally zero or a small, negative number while the match score is a positive number, larger in magnitude. This

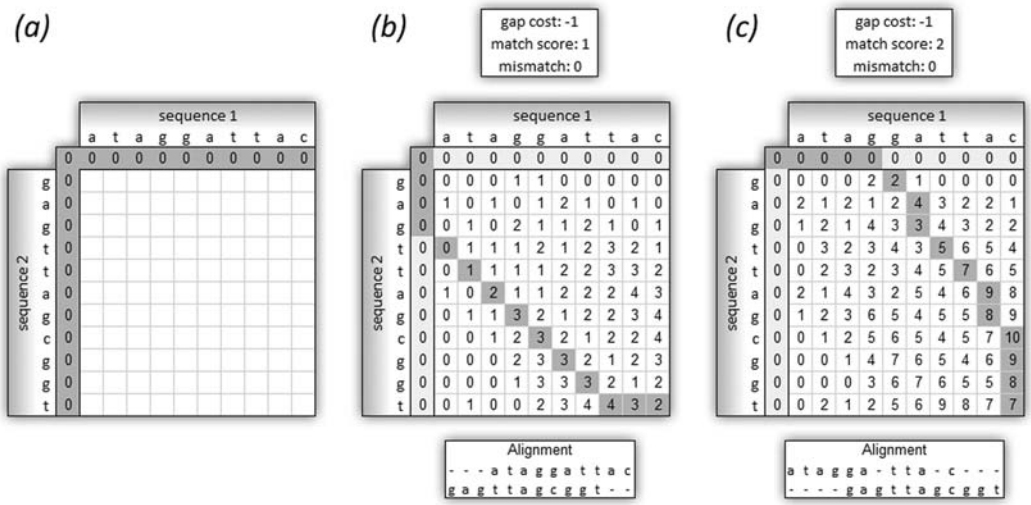


Fig. 1.2. Smith-Waterman local alignment example. (**A**) shows an empty matrix, initialized for a Smith-Waterman alignment. (**B**) and (**C**) are alignments calculated using the specified scoring parameters. The alignment produced in (**B**) is drastically different from that in (**C**), though they only differ slightly in their scoring parameters, one using a match score of 1 and the other 2.

method is used recursively, starting from the upper left corner of the matrix and proceeding to the lower right corner. **Figure 1.2b** and **c** shows matrices from two different sets of gap and match scores.

To find a local alignment, one simply finds the largest number in the matrix and traces a path back until a zero is reached, each step moving to a cell that was responsible for the current cell's value. While this method is robust and is guaranteed to give the best alignment(s) for a given set of scores and penalties, it is important to note that often multiple paths and therefore multiple alignments are possible for any given matrix when these parameters are used. As an example, **b** and **c** of **Fig. 1.2** only differ slightly in their gap and match scores, but produce very different alignments. In addition, the set of scores and penalties used dramatically affect the alignment, and finding the optimal set is neither trivial nor deterministic. Weight matrices for protein-coding sequences were developed in the late 1970s in an attempt to overcome these challenges.

2.3. Weighting/Models

2.3.1. PAM Matrices

To increase the specificity of alignment algorithms and provide a means to evaluate their statistical significance, it was necessary to implement a meaningful scoring scheme for nucleotide and amino acid substitutions. This was especially true when dealing with protein (or protein-coding) sequences. In 1978, Dayhoff et al. developed the first scoring or weighting matrices created from substitutions that have been observed during evolutionary history *(13)*. These substitutions, since they have been allowed or accepted by natural

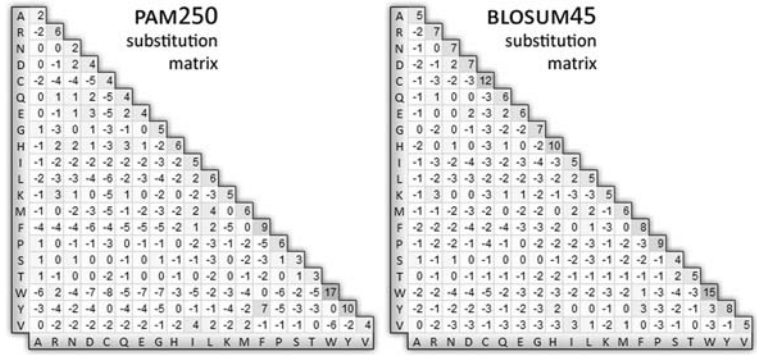


Fig. 1.3. PAM250 and BLOSUM45 substitution matrices.

selection, are called accepted point mutations (PAM). For Dayhoff's PAM matrices, groups of proteins with 85% or more sequence similarity were analyzed and their 1,571 substitutions were cataloged. Each cell of a PAM matrix corresponds to the frequency in substitutions per 100 residues between two given amino acids. This frequency is referred to as one PAM unit. Back in the 1970s, when they were created, however, there was a limited number and variety of protein sequences available, so they are biased toward small, globular proteins. It is also important to note that each PAM matrix corresponds to a specific evolutionary distance and that each is simply an extrapolation of the original. For example, a PAM250 (**Fig. 1.3**) matrix is constructed by multiplying the PAM1 matrix by itself 250 times and is viewed as a typical scoring matrix for proteins that have been separated by 250 million years of evolution.

2.3.2. BLOSUM Matrices

To overcome some of the drawbacks of PAM matrices, Henikoff and Henikoff developed the BLOSUM matrices in 1992 *(14)*. These matrices were based on the BLOCKS database, which organizes proteins into blocks, where each block, defined by an alignment of motifs, corresponds to a family. Whereas the original PAM matrix was calculated with proteins with at least 85% identity, BLOSUM matrices are each calculated separately using conserved motifs at or below a specific evolutionary distance. This diversity of matrices coupled with being based on larger datasets makes the BLOSUM matrices more robust at detecting similarity at greater evolutionary distances and more accurate, in many cases, at performing local similarity searches *(15)*.

2.3.3. Choosing a Matrix

When choosing a matrix, it is important to consider the alternatives. Do not simply choose the default setting without some initial consideration. In general, finding similarity at increasing divergence corresponds to increasing PAM matrices (PAM1, PAM40, PAM120, etc.) and decreasing BLOSUM matrices (BLOSUM90,

Table 1.3
Suggested uses for common substitution matrices. The matrices highlighted in bold are available through NCBI's BLAST web interface. BLOSUM62 has been shown to provide the best results in BLAST searches overall due to its ability to detect large ranges of similarity. Nevertheless, the other matrices have their strengths. For example, if your goal is to only detect sequences of high similarity to infer homology within a species, the PAM30, BLOSUM90, and PAM70 matrices would provide the best results. This table was adapted from results obtained by David Wheeler *(16)*

Alignment size	Best at detecting	Similarity (%)	PAM	BLOSUM
Short	Similarity within a **species**	75–90	**PAM30**	BLOSUM95
"	Similarity within a **genus**	60–75	**PAM70**	BLOSUM85
Medium	Similarity within a **family**	50–60	PAM120	**BLOSUM80**
"	The **largest range** of similarity	40–50	PAM160	**BLOSUM62**
Long	Similarity within a **class**	30–40	PAM250	**BLOSUM45**
"	Similarity within the **twilight zone**	20–30		BLOSUM30

BLOSUM80, BLOSUM62, etc.) *(16)*. PAM matrices are strong at detecting high similarity due to their use of evolutionary information. However, as evolutionary distance increases, BLOSUM matrices are more sensitive and accurate than their PAM counterparts. **Table 1.3** includes a list of suggested uses.

2.4. BLAST Programs

Nucleotide–nucleotide searches are beneficial because no information is lost in the alignment. When a codon is translated from nucleotides to amino acid, approximately 69% of the complexity is lost (64 possible nucleotide combinations mapped to 20 amino acids). In contrast, however, the true physical relationship between two coding sequences is best captured in the translated view. Matrices that take into account physical properties, such as PAM and BLOSUM, can be used to add power to the search. Additionally, in a nucleotide search, there are only four possible character states compared to 20 in an amino acid search. Thus the probability of a match due to chance versus a match due to common ancestry (identify in state versus identical by descent) is high.

The Basic Local Alignment and Search Tools (BLAST) are the most widely used and among the most accurate in detecting sequence similarity *(17)*(*see* **Note 2**). The standard BLAST programs are Nucleotide BLAST (blastn), Protein BLAST (blastp), blastx, tblastn, and tblastx. Others have also been developed to meet specific needs. When choosing a BLAST program, it is

important to choose the correct one for your question of interest. Some of the most common mistakes in similarity searching come from misunderstandings of these different applications.

- **Nucleotide blast**: compares a nucleotide query against a nucleotide sequence database
- **Protein blast**: compares a protein query against a protein sequence database
- **blastx**: compares a nucleotide query translated in all six reading frames against a protein database
- **tblastn**: compares a protein query against a nucleotide sequence database dynamically translated in all six reading frames
- **tblastx**: compares a nucleotide query in all six reading frames against a nucleotide sequence database in all six reading frames

The BLAST algorithm is an heuristic program, one that is not guaranteed to return the best result. It is, however, quite accurate. BLAST works by first making a look-up table of all the "words" and "neighboring words" of the query sequence. Words are short subsequences of length *W* and neighboring words are words that are highly accepted in the scoring matrix sense, determined by a threshold *T*. The database is then scanned for the words and neighboring words. Once a match is found, extensions with and without gaps are initiated there both upstream and downstream. The extension continues, adding gap existence (initiation) and extension penalties, and match and mismatch scores as appropriate as in the Smith-Waterman algorithm until a score threshold *S* is reached. Reaching this mark flags the sequence for output. The extension then continues until the score drops by a value *X* from the maximum, at which point the extension stops and the alignment is trimmed back to the point where the maximum score was hit. Understanding this algorithm is important for users if they are to select optimal parameters for BLAST. The interaction between the parameters *T*, *W*, *S*, *X*, and the scoring matrix allows the user to find a balance between sensitivity and specificity, alter the running time, and tweak the accuracy of the algorithm. The interactions among these variables will be discussed in **Section 2.8**.

2.5. Query Sequence

Query sequences may be entered by uploading a file or entering one manually in the text box provided (**Fig. 1.4**). The upload option accepts files containing a single sequence, multiple sequences in FASTA format, or a list of valid sequence identifiers (accession numbers, GI numbers, etc.). In contrast to previous versions of BLAST on the NCBI website, the current version allows the user to specify a descriptive job title. This allows the user to track any adjustments or versions of a search as well as its purpose and query information. This is especially important when sequence identifiers are not included in the uploaded file.

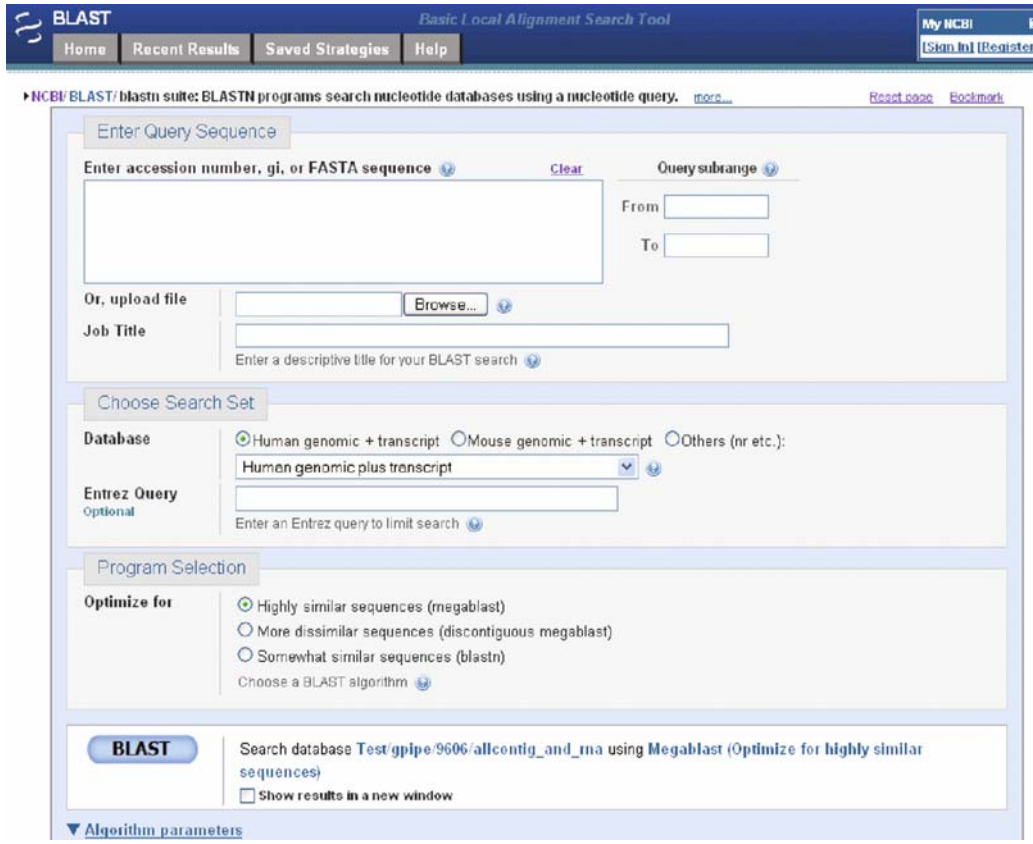


Fig. 1.4. NCBI nucleotide BLAST interface.

2.6. Search Set

2.6.1. Databases

When choosing a database, it is important to understand their purpose, content, and limitations. The list of nucleotide databases is divided into *Genomic plus Transcript* and *Other Databases* sections. Some of the databases, composed of reference sequences, come from the RefSeq database, a highly curated, all-inclusive, non-redundant set of INSDC (EMBL + GenBank + DDBJ) DNA, mRNA, and protein entries. RefSeq sequences have accession numbers of the form AA_######, where AA is one of the following combination of letters (**Table 1.4**) and ###### is a unique number representing the sequence.

A description of the nucleotide databases is included below. A list of protein databases accessible through BLAST's web interface can be found at http://www.ncbi.nlm.nih.gov/BLAST/blastcgihelp.shtml.

- **Human genomic plus transcript**: contains all human genomic and RNA sequences.
- **Mouse genomic plus transcript**: contains all mouse genomic and RNA sequences.

Table 1.4
RefSeq categories

Experimentally determined and curated		Genome annotation (computational predictions from DNA)	
NC	**C**omplete genomic molecules		
NG	Incomplete **g**enomic region		
NM	**m**RNA	XM	Model **m**RNA
NR	**R**NA (non-coding)		
NP	**P**rotein	XP	Model **p**rotein

- **Nucleotide collection (nr/nt)**: contains INSDC + RefSeq nucleotides + PDB sequences, not including EST, STS, GSS, or unfinished HGT sequences. The nucleotide collection is the most comprehensive set of nucleotide sequences available through BLAST.
- **Reference mRNA sequences (refseq_rna)**: contains the non-redundant RefSeq mRNA sequences.
- **Reference genomic sequences (refseq_genomic)**: contains the non-redundant RefSeq genomic sequences.
- **Expressed sequence tags (est)**: contains short, single reads from mRNA sequencing (via cDNA). These cDNA sequences represent the mRNA in a cell at a particular moment in a particular tissue.
- **Non-human, non-mouse ESTs (est_others)**: the previous database with human and mouse sequences removed.
- **Genomic survey sequences (gss)**: contains random genomic sequences obtained from single-pass genome surveys, cosmids, BACs, YACs, and other survey methods. Their quality varies.
- **High-throughput genomic sequences (HTGS)**: contains sequences obtained from high-throughput genome centers. Sequences in this database contain a phase number, 0 being the initial phase and 3 being the finished phase. Once finished, the sequences move to the appropriate division in their respective database.
- **Patent sequences (pat)**: contains sequences from the patent offices at each of the INSDC organizations.
- **Protein data bank (pdb)**: the nucleotide sequences from the Brookhaven Protein Data Bank managed by the Research Collaboratory for Structural Bioinformatics (http://www.rcsb.org/pdb).

- **Human ALU repeat elements (alu_repeats)**: contains a set of ALU repeat elements that can be used to mask repeat elements from query sequences. ALU sequences are regions subject to cleavage by Alu restriction endonucleases, around 300 bp long, and estimated to constitute about 10% of the human genome *(18)*.
- **Sequence tagged sites (dbsts)**: a collection of unique sequences used in PCR and genome mapping that identify a particular region of a genome.
- **Whole-genome shotgun reads (wgs)**: contains large-scale shotgun sequences, mostly unassembled and non-annotated.
- **Environmental samples (env_nt)**: contains sets of whole-genome shotgun reads from many sampled organisms, each set from a particular location of interest. These sets allow researchers to look into the genetic diversity existing at a particular location and environment.

2.6.2. Organism

The organism box allows the user to specify a particular organism to search. It automatically suggests organisms when you begin typing. This option is not available when Genomic plus Transcript databases are selected (**Fig. 1.5**).

2.6.3. Entrez Queries

Entrez queries provide a way to limit your search to a specific type of organism or molecule. It is an efficient way to filter unwanted results by excluding organisms or defining sequence length criteria. In addition, Entrez queries allow the user to find sequences submitted by a particular author, from a particular journal, with a particular property or feature key, or submitted or modified within a specific date range. For help with Entrez queries, see the Entrez Help document at http://www.ncbi.nlm.nih.gov/entrez/query/static/help/helpdoc.html.

2.7. BLAST Search Parameters

In addition to entering a query sequence, choosing a search set, and selecting a program, several additional parameters are available, which allow you to fine-tune your search to your needs. These parameters are available by clicking the "Algorithm parameters" link at the bottom of the BLAST page (**Fig. 1.6**) (*see* **Notes 3 and 4**).

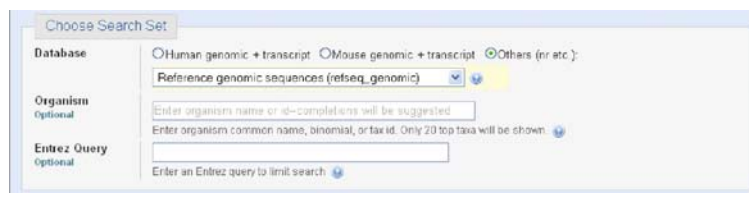


Fig. 1.5. NCBI nucleotide BLAST algorithm parameters.

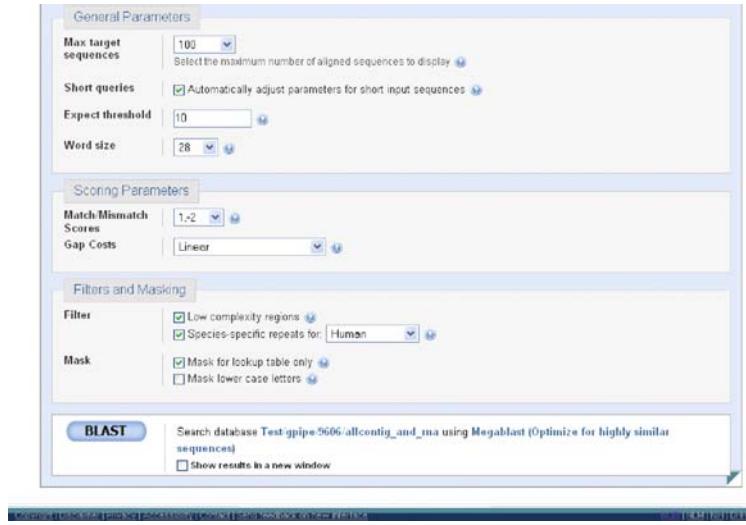


Fig. 1.6. Organism selection when searching a multi-organism database.

2.7.1. Max Target Sequences

The maximum target sequences parameter allows you to select the number of sequences you would like displayed in your results. Lower numbers do not reduce the search time, but do reduce the time to send the results back. This is generally only an issue over a slow connection.

2.7.2. Short Queries

When using short queries (of length 30 or less), the parameters must be adjusted or you will not receive statistically significant results. Checking the "short queries" box automatically adjusts the parameters to return valid responses for a short query sequence.

2.7.3. Expect Threshold

The expect threshold limits the results displayed to those with an *E*-value lower than it. This value corresponds to the number of sequence matches that are expected to be found merely by chance.

2.7.4. Word Size

The word size, *W*, as discussed earlier determines the length of the words and neighboring words used as initial search queries. Increasing the word size generally results in fewer extension initializations, increasing the speed of the BLAST search but decreasing its sensitivity.

2.7.5. Scoring Parameters

The scoring parameters of a nucleotide search are the match and mismatch scores and gap costs. In protein searches, the match and mismatch scores are indicated by a scoring matrix (*see* **Section 2.3**). A limited set of suggested match and mismatch scores are available from the dropdown menu on NCBI's BLAST search form. Increasing the ratio in the following fashion (match, mismatch): (1,–1) → (4,–5) → (2,–3) → (1,–2) → (1,–3) → (1,–4) prevents mismatched nucleotides from aligning, increasing the

Table 1.5
Suggested scoring parameters for nucleotide–nucleotideBLAST searches. When performing a nucleotide–nucleotide BLAST search, these general guidelines may be used to choose a match/mismatch score based upon the degree of conservation you expect to see in your results. If you are searching for sequences with a high degree of similarity (i.e., within a species), the default parameters of (match +1, mismatch –2) would be appropriate. If, however, you are searching for sequences between very distant organisms (a worm and a mouse, for example), a smaller ratio would be more appropriate (for example, –1). Information provided by NCBI *(26)*

Match/mismatch ratio	Similarity (%)
0.33 (1/–3)	99
–0.5 (1/–2)	95
–1 (1/–1)	75

number of gaps, but decreasing mismatches. The greater divergence you expect in sequences you are looking for, the larger the ratio you should choose. NCBI has provided the guidelines found in **Table 1.5**. Additionally, decreasing the gap existence and extension penalties will increase gap incidence.

2.7.6. Filters

The low complexity regions filter removes regions of the sequence with low complexity, preventing those segments from producing statistically significant but uninformative results. The DUST program by Tatusov and Lipman (unpublished) is used for nucleotide BLAST searches. Often, when a search takes much longer than expected, the query contains a low-complexity region that is being matched with many similar but unrelated sequences. It is important to note, however, that turning this filter on may remove some interesting and informative matches from the results. In nucleotide searches, it is also possible to remove species-specific repeats by checking the "Species-specific repeats for:" box and selecting the appropriate species. This prevents repeats that are common in a particular species from producing false-positives with other parts of its own or closely related genomes.

2.7.7. Masks

The "Mask for lookup table only" option allows the user to mask the low-complexity regions (regions of biased composition including homopolymeric runs, short-period repeats, etc.) during the

seeding stage, where words and neighboring words are scanned, but unmask them during the extension phases. This prevents the *E*-values from being affected in biologically interesting results while preventing regions of low complexity from slowing the search down and introducing uninteresting results.

The "Mask lower case letters" option gives the user the option to annotate his or her sequence by using lower case letters where masking is desired.

2.8. Interpreting the Results

By default, BLAST results contain five basic sections: a summary of your input (query and parameters), a graphical overview of the top results, a table of sequences producing significant alignments, the best 100 alignments, and result statistics. The number of hits shown in the graphical overview as well as the number of alignments, among other options, may be changed by clicking "Reformat these results" at the top of the results page or by clicking "Formatting options" on the Formatting Results page (the page that appears after you click BLAST and before the results appear).

In the third section, the results table contains eight columns: accession, description, max score, total score, query coverage, E-value, max ident, and links. The *Accession* number provides a link to detailed information about the sequence. The *description* provides information about the species and the kind of sample the hit was generated from. The *max score* provides a metric for how good the best local alignment is. The *total score* indicates how similar the sequence is to the query, accounting for all local alignments between the two sequences. If the max score is greater than the total score, then more than one local alignment was found between the two sequences. Higher scores are correlated with more similar sequences. Both of these scores, reported in bits, are calculated from a formula that takes into account matches (or similar residues, if doing a protein search) and mismatch penalties along with gap insertion penalties. Bit scores are normalized so that they can be directly compared even though the alignments between different sequences may be of different lengths. The expectation value or *E-value* provides an estimate of how likely it is that this alignment occurred by random chance. An E-value of 2e–02 indicates that similarity found in the alignment has a 2 in 100 chance of occurring by chance. The lower the E-value, the more significant the score. An appropriate cutoff E-value depends on the users' goals. The *max identity* field shows the percentage of the query sequence that was identical to the database hit. The *links* field provides links to UniGene, the Gene Expression Omnibus, Entrez Gene, Entrez's Related Structures (for protein sequences), and the Map Viewer (for genomic sequences).

2.9. Future of Similarity Searching

Since both PAM and BLOSUM matrices are experimentally derived from a limited set of sequences in a database that was available at the time they were created, they will almost certainly not provide optimal values for searches with new sequence

families. Current research is being performed to determine which chemical properties are changing in a sequence in order to provide a magnitude of change that is independent of scoring matrices.

Current techniques to find promoter regions are severely lacking in accuracy *(19)*. Techniques will arise in the future that may improve current methods by using BLAST-like algorithms to assess the similarity of a sequence to known promoter elements, thus helping to identify it as a promoter.

3. Examples

This section will provide three examples of common BLAST uses: a nucleotide–nucleotide BLAST, a position-specific iterated BLAST, and a blastx.

3.1. Nucleotide–Nucleotide BLAST for Allele Finding

Here we present an example of using BLAST to search for the known alleles of a given nucleotide sequence. This approach can be used to answer the question: what are the known variants of my gene of interest (within its species)? Our example will be to find all known variants of a Tp53 nucleotide sequence (accession number AF151353) from a mouse. While this sequence does code for a protein, non-coding sequences would work just as well using this approach.

We will start by going to the BLAST homepage at http://www.ncbi.nlm.nih.gov/BLAST/ and selecting *nucleotide blast*. In the "Enter Query Sequence" box, we type the accession number: AF151353. You will notice that the "Job Title" box automatically fills in a title for you "AF151353:*Mus musculus* tumor suppressor p53...". If we were to paste a sequence instead of an accession number or GI, we would want to enter a job title to help us keep track of our results. Under "Choose Search Set," we select the "Nucleotide collection (nr/nt)" database, since it is the most comprehensive database (remember that nr is no longer non-redundant). For a complete search, we should also perform a search on the "Expressed sequence tags (est)" database. In the Organism box, we choose type "mouse" and select "mouse (taxid:10090)," which corresponds to *Mus musculus*, the house mouse. Since we are searching for alleles, we select "Highly similar sequences (megablast)" in the "Program Selection" box.

Next, let us change the algorithm parameters. Click "Algorithm parameters" to display them. Since the sequence is 1,409 bp in length, we deselect the "Automatically adjust parameters for short input sequences" box. Since we expect that the p53 protein is a well-conserved protein (due to its critical function), we set the expect threshold to a low value. Let us choose 1e-8. For a word

size, we are not concerned about speed in this case, so the number of extensions performed is not a concern. Let us select a word size of 20 to make sure we do not miss any matches (although in this case a larger word size should not make much difference). As for the scoring parameters, we choose the largest ratio, corresponding to the greatest identity: "1,–4." Since this is a protein-coding sequence, we do not expect repeats to be a factor, so we leave the Filters and Masking section at the default settings.

The results indicate that 108 hits were found on the query sequence. Looking at the graphical alignment (**Fig. 1.7**), we notice that only about 2/3 of them span a good portion of the query. When we scroll down to the gene descriptions, most of the last fourth are pseudogenes (partial sequence) (**Fig. 1.8**), which may offer insight into different alleles and their corresponding phenotypes, but which were not sequenced experimentally. Performing a search on the EST database with the same parameters results in 101 additional hits.

3.2. PSI-BLAST for Distant Homology Searching

When searching for distantly related sequences, two BLAST options are available. One is the standard nucleotide–nucleotide BLAST with discontiguous BLAST, a method very similar to Ma

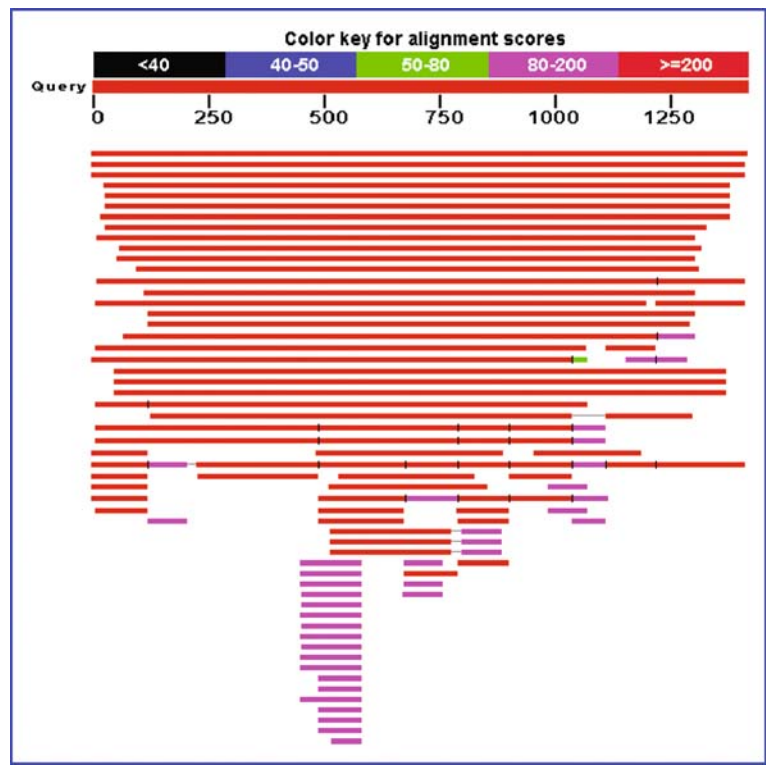
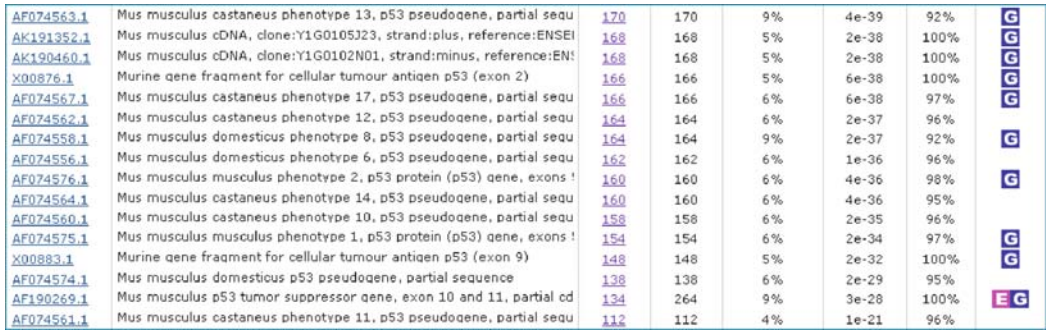


Fig. 1.7. Graphical distribution of top 100 BLAST hits.



AF074563.1	Mus musculus castaneus phenotype 13, p53 pseudogene, partial sequ	170	170	9%	4e-39	92%	G
AK191352.1	Mus musculus cDNA, clone:Y1G0105J23, strand:plus, reference:ENSEI	168	168	5%	2e-38	100%	G
AK190460.1	Mus musculus cDNA, clone:Y1G0102N01, strand:minus, reference:EN	168	168	5%	2e-38	100%	G
X00876.1	Murine gene fragment for cellular tumour antigen p53 (exon 2)	166	166	5%	6e-38	100%	G
AF074567.1	Mus musculus castaneus phenotype 17, p53 pseudogene, partial sequ	166	166	6%	6e-38	97%	G
AF074562.1	Mus musculus castaneus phenotype 12, p53 pseudogene, partial sequ	164	164	6%	2e-37	96%	
AF074558.1	Mus musculus domesticus phenotype 8, p53 pseudogene, partial sequ	164	164	9%	2e-37	92%	G
AF074556.1	Mus musculus domesticus phenotype 6, p53 pseudogene, partial sequ	162	162	6%	1e-36	96%	
AF074576.1	Mus musculus musculus phenotype 2, p53 protein (p53) gene, exons !	160	160	6%	4e-36	98%	G
AF074564.1	Mus musculus castaneus phenotype 14, p53 pseudogene, partial sequ	160	160	6%	4e-36	95%	
AF074560.1	Mus musculus castaneus phenotype 10, p53 pseudogene, partial sequ	158	158	6%	2e-35	96%	
AF074575.1	Mus musculus musculus phenotype 1, p53 protein (p53) gene, exons !	154	154	6%	2e-34	97%	G
X00883.1	Murine gene fragment for cellular tumour antigen p53 (exon 9)	148	148	5%	2e-32	100%	G
AF074574.1	Mus musculus domesticus p53 pseudogene, partial sequence	138	138	6%	2e-29	95%	
AF190269.1	Mus musculus p53 tumor suppressor gene, exon 10 and 11, partial cd	134	264	9%	3e-28	100%	E G
AF074561.1	Mus musculus castaneus phenotype 11, p53 pseudogene, partial sequ	112	112	4%	1e-21	96%	

Fig. 1.8. Last 16 sequences producing significant alignments from a mouse p53 gene Nucleotide BLAST search. Nineteen of the last 26 reported sequences are pseudogenes.

et al.'s work *(20)*, selected as the program. The other is to use a more sensitive approach, PSI-BLAST, which performs an iterative search on a protein sequence query. Though the second approach will only work if you are dealing with protein-coding sequences, it is more sensitive and accurate than the first.

In this example, we will search for relatives of the cytochrome *b* gene of the Durango night lizard (*Xantusia extorris*). We start by selecting *protein blast* from the BLAST home page and entering the accession number, ABY48155, into the query box. If your sequence is not available as a protein sequence, you will need to translate it. This can easily be done using a program such as MEGA *(21)*, available at http://www.megasoftware.net, or an online tool such as the JustBio Translator (http://www.justbio.com/translator/) or the ExPASy Translate Tool (http://www.expasy.org/tools/dna.html).

Once again, the "Job Title" box is filled with "ABY48155: cytochrome b [*Xantusia extorris*]." We will choose the "Reference proteins (refseq_protein)" database, which is more highly curated and non-redundant (per gene) than the default nr database. We do not specify an organism because we want results from any and all related organisms. For the algorithm, we select PSI-BLAST due to its ability to detect more distantly related sequences. We hope to include as many sequences as possible in our iterations, so we choose 1,000 as the max target sequences. We can, once again, remove the "Automatically adjust parameters for short input sequences" check, since our sequence is sufficiently long (380 amino acids). Since we wish to detect all related sequences, we keep the expect threshold at its default of 10. While decreasing it may remove false-positives, it may also prevent some significant results from being returned. Since we do not have a particular scope in mind (within the genus or family, for example), we will use the BLOSUM62 matrix due to its ability to detect homology over large ranges of similarity.

The first iteration results in 1,000 hits on the query sequence, all of which cover at least 93% of the query sequence and have an E-value of 10^{-126} or less. We leave all of the sequences selected and press the "Run PSI-Blast iteration 2" button. The second iteration likewise returns 1,000 hits, but this time they have E-values less than 10^{-99} and cover at least 65% of the query sequence (all but six cover 90% or more). We uncheck the last hit, Bi4p [*Saccharomyces cerevisiae*], since we are unsure of its homology, and iterate one last time.

At this point, it would be helpful to view the taxonomy report of the results. You can do so by clicking "Taxonomy Reports" near the bottom of the first section of the BLAST report. You will notice that we have a good selection of organisms, ranging from bony fishes to Proteobacteria. While this list would need to be narrowed to produce a good taxonomy, it would be a good starting point if you wish to perform a broad phylogenetic reconstruction. To perform a search of more closely related sequences, you would likely perform a standard blastp (protein–protein BLAST) instead of a PSI-BLAST and use the PAM 70 or PAM 30 matrix.

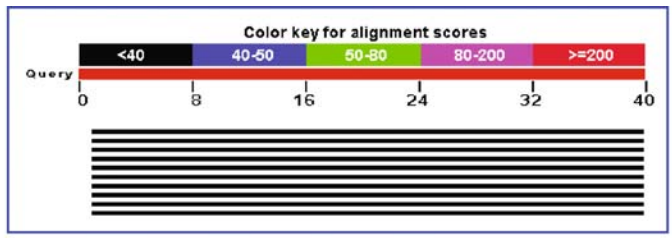
3.3. Blastx for EST Identification

What if you have a nucleotide sequence such as an expressed sequence tag and wish to know if it codes for a known protein? You can search the nucleotide database or take the more direct approach of blastx. Blastx allows you to search the protein database using a nucleotide query, which it first translates into all six reading frames. In this example, we will perform a blastx on the following sequence:

TCTCTATAGTTATGGTGTTCTGAATCAGCCTTCCCTCATA

Since the sequence is only 40 bp long, we need to be careful with our parameters. We start by selecting blastx from the BLAST homepage. We then enter the sequence into the query box and enter a relevant job title, such as "EST blastx Search 1." We will search the "Non-redundant protein sequences (nr)" database, since it has the largest number of annotated nucleotide sequences. Under "Algorithm parameters," we need to choose an appropriate expect threshold and matrix. If we choose too low an expect threshold, we might not find anything. Likewise, if we choose the wrong matrix, we may not obtain significant results due to the short length of our sequence. We will choose 10 (the default) as our expect threshold and PAM70 as our matrix, since it corresponds to finding similarity at or below the family/genus level. Since we do not know what our sequence is, we want to filter regions of low complexity to ensure that if our sequence contains such regions, they will not return deceptively significant results.

Our search produces a large number (more than 1,000) of results with an E-value of 0.079 (**Fig. 1.9**). If we were to use the PAM70 matrix, essentially the same results would be obtained, but each with an E-value of 3.0. Since all of the 2,117 results are different entries of the nucleocapsid protein of the Influenza A



Legend for links to other resources: U UniGene E GEO G Gene S Structure M Map Viewer

Sequences producing significant alignments:
(Click headers to sort columns)

Accession	Description	Max score	Total score	Query coverage	E value	Max ident	Links
ABY81430.1	nucleocapsid protein [Influenza A virus (A/swine/Iowa/1/1986(H1N1))	38.3	38.3	97%	0.079	92%	
ABO43788.1	nucleocapsid protein [Influenza A virus (A/mallard/ON/499/2005(H5N	38.3	38.3	97%	0.079	92%	
CAN89845.1	nucleoprotein [Influenza A virus (A/wild boar/Germany/R169/2006(H	38.3	38.3	97%	0.079	92%	
ABO12377.1	nucleocapsid protein [Influenza A virus (A/mallard/Manitoba/458/200	38.3	38.3	97%	0.079	92%	
ABV53582.1	nucleoprotein [Influenza A virus (A/chicken/Nigeria/1071-30/2007(H5	38.3	38.3	97%	0.079	92%	
ABV53572.1	nucleoprotein [Influenza A virus (A/chicken/Nigeria/1071-29/2007(H5	38.3	38.3	97%	0.079	92%	
ABV53532.1	nucleoprotein [Influenza A virus (A/chicken/Nigeria/1071-10/2007(H5	38.3	38.3	97%	0.079	92%	
ABV53512.1	nucleoprotein [Influenza A virus (A/chicken/Nigeria/1071-7/2007(H5N	38.3	38.3	97%	0.079	92%	
ABV53492.1	nucleoprotein [Influenza A virus (A/chicken/Nigeria/1071-4/2007(H5N	38.3	38.3	97%	0.079	92%	
ABV53482.1	nucleoprotein [Influenza A virus (A/chicken/Nigeria/1071-3/2007(H5N	38.3	38.3	97%	0.079	92%	

Fig. 1.9. Blastx results showing E-values of 0.079 for the top ten<10> hits, all of which are nucleocapsid proteins or nucleoproteins.

virus, we can be somewhat confident that our protein is related, especially if we had any prior knowledge that would support our findings.

4. Notes

1. One of the options NCBI provides from their homepage is to search across their databases using an identifier (accession number, sequence identification number, Locus ID, etc.). This option can be rather straightforward if you are using an identifier unique to a particular sequence; however, if you are searching for a locus across organisms or individuals, you may need to pay close attention to the search terms you are using. For example, since the Cytochrome b/b6 subunit is known by the terms "Cytochrome b," "Cytochrome b6," "cyt-b," "cytb," "cyb," "COB," "COB1," "cyb6," "petB," "mtcyb," and "mt-cyb" in a search for all possible homologs of this subunit, it is necessary to search for all of its names and abbreviations used in the organisms of interest. Since research groups studying different organisms create their own unique locus names for the same gene, it is important to use all of them in your search. IHOP (www.ihop-net.org) is an excellent resource for protein names *(22)*. In addition, you will want to perform a BLAST search to make sure you have everything!

Fig. 1.10. Save search strategies.

2. In addition to the BLAST program provided by NCBI, other BLAST programs exist, which have improved the BLAST algorithm in various ways. Dr. Warren Gish at Washington University in St. Louis has developed WU-BLAST, the first BLAST algorithm that allowed gaped alignments with statistics *(23)*. It boasts speed, accuracy, and flexibility, taking on even the largest jobs. Another program, FSA-BLAST (Faster Search Algorithm), was developed to implement recently published improvements to the original BLAST algorithm *(24)*. It promises to be twice as fast as NCBI's and just as accurate. WU-BLAST is free for academic and non-profit use and FSA-BLAST is an open source under the BSD license agreement.
3. My NCBI is a tool that allows you to customize your preferences, save searches, and set up automatic searches that send results via e-mail. If you find yourself performing the same searches (or even similar searches) repeatedly, you may want to take advantage of this option! To register, go to the NCBI home page and click the "My NCBI" link under "Hot Spots." Once you have registered and signed in, a new option will be available to you on all BLAST and Entrez searches (**Fig. 1.10**).
4. To save a BLAST search strategy, simply click the "Save Search Strategies" link on the results page. This will add the search to your "Saved Strategies" page, which is available through a tab on the top of each page in the BLAST website when you are logged in to My NCBI. Doing so will not save your results, but it will save your query and all parameters you specified for your search so you can run it later to retrieve updated results.

References

1. Dayhoff, M. O., Eck, R. V., Chang, M. A., and Sochard, M. R. (1965) Atlas of Protein Sequence and Structure, National Biomedical Research Foundation, Silver Spring, MD.
2. Hersh, R. T. (1967) Reviews. *Syst Zool* **16,** 262–63.
3. Galperin, M. Y. (2007) The molecular biology database collection: 2007 update. *Nucleic Acids Res* **35,** D3–D4.
4. Batemen, A. (2007) Editorial. *Nucleic Acids Res* **35,** D1–2.
5. Pearson, W. R., and Lipman, D. J. (1988) Improved tools for biological sequence comparison. *Proc Natl Acad Sci USA* **85,** 2444–48.
6. León, D., and Markel, S. (2003) *Sequence Analysis in a Nutshell*, O'Reilly & Associates, Inc., Sebastopol, CA.

7. Sample GenBank Record [Internet]. National Library of Medicine, Bethesda, MD; [modified October 23, 2006; cited November 24, 2007]. Available from: http://www.ncbi.nlm.nih.gov/Sitemap/samplerecord.html
8. EMBL Nuleotide Sequence Database User Manual [Internet]. The European Bioinformatics Institute, Cambridge, United Kingdom; [modified June 7, 2007; cited November 24, 2007]. Available from: http://www.ebi.ac.uk/embl/Documentation/User_manual/usrman.html
9. Explanation of DDBJ flat file Format [Internet]. DNA Data Bank of Japan, Mishima, Shizuoka, Japan; [modified August 7, 2007; cited November 24, 2007]. Available from: http://www.ddbj.nig.ac.jp/sub/ref10-e.html
10. NCBI-GenBank Flat File Release 162.0 [Internet]. National Library of Medicine; [modified October 15, 2007; cited November 20, 2007]. Available from: ftp://ftp.ncbi.nih.gov/genbank/gbrel.txt
11. Needleman, S. B., and Wunsch, C. D. (1970) A general method applicable to the search for similarities in the amino acid sequence of two proteins. *J Mol Biol* **48,** 443–53.
12. Smith, T. F., and Waterman, M. S. (1981) Identification of common molecular subsequences. *J Mol Biol* **147,** 195–97.
13. Dayhoff, M. O., Schwartz, R. M., and Orcutt, B. C. (1978) *Atlas of Protein Sequence and Structure* (Foundation, N. B. R., Ed.), Vol. 5, pp. 345–58, National Biomedical Research Foundation., Silver Spring, MD.
14. Henikoff, S., and Henikoff, J. G. (1992) Amino Acid Substitution Matrices from Protein Blocks. *Proc Natl Acad Sci USA* **89,** 10915–19.
15. Baxevanis, A. D., and Ouellette, B. F. F. (2005) Bioinformatics: A Practical Guide to the Analysis of Genes and Proteins, John Wiley & Sons, Inc., Hoboken, New Jersey.
16. Wheeler, D. G. (2003) Selecting the right protein scoring matrix. *Curr Proto Bioinformat* 3.5.1–6.
17. Altschul, S. F., Gish, W., Miller, W., Myers, E. W., and Lipman, D. J. (1990) Basic local alignment search tool. *J Mol Biol* **215,** 403–10.
18. Roy-Engel, A. M., Carroll, M. L., Vogel, E., Garber, R. K., Nguyen, S. V., Salem, A. H., Batzer, M. A., and Deininger, P. L. (2001) Alu insertion polymorphisms for the study of human genomic diversity. *Genetics* **159,** 279–90.
19. Tompa, M., Li, N., Bailey, T. L., Church, G. M., De Moor, B., Eskin, E., Favorov, A. V., Frith, M. C., Fu, Y. T., Kent, W. J., Makeev, V. J., Mironov, A. A., Noble, W. S., Pavesi, G., Pesole, G., Regnier, M., Simonis, N., Sinha, S., Thijs, G., van Helden, J., Vandenbogaert, M., Weng, Z. P., Workman, C., Ye, C., and Zhu, Z. (2005) Assessing computational tools for the discovery of transcription factor binding sites. *Nat Biotechnol* **23,** 137–44.
20. Ma, B., Tromp, J., and Li, M. (2002) PatternHunter: faster and more sensitive homology search. *Bioinformatics* **18,** 440–5.
21. Tamura, K., Dudley, J., Nei, M., and Kumar, S. (2007) MEGA4: molecular evolutionary genetics analysis (MEGA) software version 4.0. *Mol Biol Evol* **24,** 1596–9.
22. Hoffmann, R., and Valencia, A. (2004) A gene network for navigating the literature. *Nat Genet* **36,** 664–64.
23. Gish, W. (1996–2004) WU BLAST 2.0 [Internet]. Saint Louis, MO; [modified March 22, 2006; cited January 3, 2008]. Available from: http://blast.wustl.edu
24. Cameron, M., Williams, H. E., Bernstein, Y., and Cannane, A. (2004–2006) FSA BLAST [Internet]. [modified March 8, 2006; cited January 3, 2008]. Available from: http://www.fsa-blast.org
25. Madden, T. (2002) The BLAST Sequence Analysis Tool [Internet]. National Library of Medicine, Bethesda, MD; [modified August 13, 2003; cited January 4, 2008]. Available from: http://www.ncbi.nlm.nih.gov/books/bv.fcgi?rid=handbook
26. Web BLAST page options [Internet]. National Library of Medicine, Bethesda, MD; [cited January 4, 2008]. Available from: http://www.ncbi.nlm.nih.gov/BLAST/blastcgihelp.shtml#Reward-penalty

Chapter 2

Gene Orthology Assessment with *OrthologID*

Mary Egan, Ernest K. Lee, Joanna C. Chiu, Gloria Coruzzi, and Rob DeSalle

Abstract

OrthologID (http://nypg.bio.nyu.edu/orthologid/) allows for the rapid and accurate identification of gene orthology within a character-based phylogenetic framework. The Web application has two functions – an orthologous group search and a query orthology classification. The former determines orthologous gene sets for complete genomes and identifies diagnostic characters that define each orthologous gene set; and the latter allows for the classification of unknown query sequences to orthology groups. The first module of the Web application, the gene family generator, uses an E-value based approach to sort genes into gene families. An alignment constructor then aligns members of gene families and the resulting gene family alignments are submitted to the tree builder to obtain gene family guide trees. Finally, the diagnostics generator extracts diagnostic characters from guide trees and these diagnostics are used to determine gene orthology for query sequences.

Key words: Single linkage cluster, orthology, phylogeny, alignment, diagnosis genomics.

Homology is either the cornerstone of biology or a term ripe for burning.
John Maynard Smith

1. Introduction

Sub-genomic studies (that analyze hundreds to thousands of gene regions for perhaps hundreds of terminal taxa) face new computational challenges. Responding to these challenges is resulting in the development of new methodologies and tools for tree building and analysis. Sub-genomic studies, however, follow the same ground plan as most molecular systematic studies, albeit on a grand scale. It may be that the next stimulus to intellectual debate in the field of systematics will come as a result of responding to

David Posada (ed.), *Bioinformatics for DNA Sequence Analysis,* Methods in Molecular Biology 537

DOI 10.1007/978-1-59745-251-9_2

challenges faced by true phylogenomic studies in which entire genomes are used as the basis for comparison. The presence of gene families and horizontal gene transfer pose challenges for both character coding as well as for identifying which are the appropriate terminals in the analysis. These challenges to be faced in phylogenomic studies hearken back to those faced in morphological studies. There is likely to be a shift in the direction of intellectual debate in the phylogenetic analysis equation, the direction of this shift being toward data matrix assembly and homology assessment. Several types of genomic studies are already implicitly or explicitly addressing the problem of homology.

Homology is often considered one of Darwin's most impressive contributions to evolutionary thinking. Darwin was one of the first to discern the differences between homology and analogy. Whereas homology refers to traits that are the same due to common ancestry, analogy refers to traits that are similar due to evolutionary convergence. Homology then becomes a term of absoluteness and must be discovered via hypothesis testing, and similarity becomes a term of measurement and is calculated from observation of two entities. The literature on homology is rich with debate, hence the quote that starts off this chapter. While phylogenomic studies may stimulate intellectual debate and growth in systematic theory, the converse may also be true – that systematics may provide an aspect of the intellectual underpinning necessary for the continued development of genomic studies.

In the 1980s Fitch and several colleagues published an important clarification of terms then being used in the earliest of comparisons of molecular sequences. Their note suggested that scientists take care in using the word homology. Essentially, Fitch et al. pointed out that most molecular biologists at the time were misusing and hence overusing the term homology. For instance, a typical sequence comparison paper from that period would claim that two sequences of 100 amino acids long had 70% homology if 70 out of 100 residues in the two proteins were the same. Fitch and colleagues pointed out that this was a misuse of the term homology, which indicates common ancestry. When two sequences are compared there is necessarily a lack of investigation of common ancestry because it takes at least three target sequences and an outgroup sequence to discover common ancestry. Those readers who have heard Fitch speak are probably familiar with his famous punch line about homology – "Homology is like pregnancy. Someone is either pregnant or not. A person cannot be 70% pregnant." In this context, two sequences can be homologous, but cannot be 70% homologous. Rather two sequences are 70% similar (for further discussion of similarity versus common ancestry in orthology assessment, *see* **Note 1**).

In addition to establishing this important distinction concerning homology, Fitch and colleagues also established a framework for how we should examine genes in multigene families, by proposing

that the term orthology refer to genes that are identical by descent as a result of speciation of two entities. The term paralogy then refers to a pair or set of genes related to each other but not through a speciation event. In this case, paralogy refers to members of a gene family that have arisen through duplication and not followed by a speciation event. While both terms refer to kinds of homologous relationships, orthology is what an anatomist would refer to as homology and paralogy would be akin to serial homology. A third term coined xenology refers to the similarity of entities as a result of horizontal transfer. It should be obvious from the terminology that any departure from orthology for members of gene families complicates and even negates sound evolutionary or biological analysis. The old adage of comparing apples to oranges also applies to genes in gene families.

To automate the rapid assessment of orthology of genes in gene families we have developed a Web based program called *OrthologID (1)*. This program uses the concept of common ancestry to establish homologous relationships of genes obtained from whole genome sequencing, EST studies, or other genome analyses. There are other methods that exist that have been used to establish orthology, such as simple BLAST, BLAT, and COG approaches. Because BLAST best hits have been shown not to identify the closest phylogenetic neighbor *(2)*, a problem exists with relying solely on this approach (and indeed others that rely on BLAST or are similar to BLAST). However, BLAST, BLAT, and other techniques such as COGs and other distance measures (*see* **Note 1**) can be informative first steps in topographical assessment. We consider these approaches to be valid generators of *hypotheses* when determining orthology and as we point out below they are incorporated into the algorithm we have developed. Our description of the program will first describe the rationale for the approach we have devised, then describe in detail the algorithm and its component parts, and finally some worked examples are presented.

An automated approximation of the phylogenetic gene tree approach to orthology determination would need to be developed in order to be able to use this approach on a genomic scale. For small gene families or for limited numbers of taxa, it is possible to use this approach with currently available analytical tools. These would involve initial similarity searches to identify putative gene families (for multiple taxa simultaneously), alignment, tree building, and screening trees for diagnostic characters to identify orthogous gene family members of each taxon, These analyses would be repeated for each new sequence to be placed among its orthologous group. To use this approach on a genomic scale, new tools have been developed. The main difference between the manual phylogenetic gene tree approach and the automated approach implemented in *OrthologID* is the exclusive use of completely sequenced genomes for constructing gene family trees. These

trees (termed guide trees) are screened for the presence of characters diagnostic of orthologs using the CAOS algorithm *(3)* and the identification of unknown queries is made through comparison of the guide tree diagnostics and the query sequence. In this way trees are not required to be constructed each time a new query's orthology is identified.

Table 2.1
Description of homology approaches showing the methods used to establish homology and focus of application

Approach and steps	Description of step	Purpose or method	Applied to
dePinna (dP)			
1. Primary homology	Establish character coding	Interpret anatomy (similar to character assignment)	Anatomy
2. Secondary homology	Phylogenetic analysis	Discover shared and derivedness to establish homology (identical to step 3 in BS)	
Brower Schawaroch (BS)			
1. Topographical similarity	Sequence alignment	New aspect not in dP	Molecular sequences
2. Character assignment	Assess character transformations	Simple for DNA and proteins to establish homology	
3. Phylogenetic analysis	Discover shared and derivedness	(Identical to step 2 in dP)	
Gene family homology (GFH)			
1. Topographical similarity	BLAST/BLAT	Establishes hypothesis of gene family inclusion	Gene presence absence studies; gene family studies
2. Character assignment	Alignment of gene	Establishes character assignment for sequences	
3. Phylogenetic analysis	Because target now is organismal phylogenetic analysis accomplishes gene homology assessment	Establishes gene family homologies by demonstrating shared derived origin for family members	

Establishing homology or orthology of genes in gene families can be viewed as slightly different from the way that DNA sequence characters are treated during alignment (for a discussion of the phylogenetic basis of homology assessment as applied to sequence alignment, *see* **Note 2**).

The *OrthologID* approach to orthology assessment is similar in many ways to the Brower and Schawaroch *(4)* scheme for homology assessment of phylogenetic characters via alignment. The differences between that approach and orthology determination is as follows: First, the potential members of gene family are identified using topographical similarity. Topographical similarity for orthology studies is very similar to the Brower and Schawaroch *(4)* scheme, except that the determination of topographical similarity is complicated by potential paralogy problems. Instead of going straight to alignment to determine topographical similarity a preliminary step of determining group membership using similarity as a means to assess group membership is implemented. The inclusion of genes into a particular gene family is often accomplished by setting a similarity cutoff (usually using similarity comparisons like E-values of sequences like BLAST *(5)*, BLAT, or COG *(6)*; *see* **Note 3**) and including all genes in an ortholog group that conform to the predetermined cutoff. Once this first step is accomplished the alignment step can be undertaken. Once the alignment step is accomplished the establishment of character state identity is straightforward and can be followed by the test of the hypothesis using phylogenetic approaches. **Table 2.1** summarizes the differences in the three ways of looking at homology.

2. Program Usage

The Web based *OrthologID* server allows the user to input a sequence or sequences of unknown orthology and receives the ortholog identification along with critical diagnostics for the ortholog groups. The *OrthologID* approach was developed specifically to handle the burgeoning amount of EST data from plant genomics. An overview of the *OrthologID (1)* approach is shown in **Fig. 2.1**. *OrthologID* uses the three step approach of orthology identification that are outlined in **Table 2.1** and discussed above. The approach is similar in some ways to PhiG developed by Dehal and Boore, (*7*; *see* **Note 4**).

The program is based on the structure and maintenance of a database that we call the *OrthologID* database (**Fig. 2.1**). To date the Plant *OrthologID* database is composed of five fully sequenced genomes – *Arabidopsis thaliana, Oryza sativa, Populus trichocarpa, Physcomitrella patens*, and *Chlamydomonas reinhardtii*,

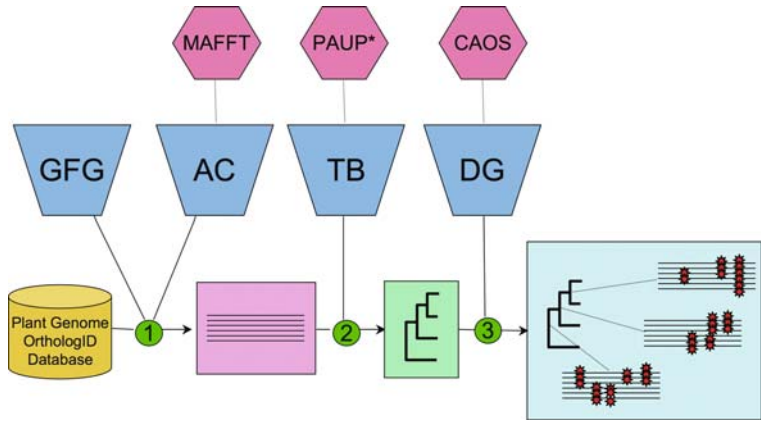


Fig. 2.1. Schematic diagram of the *OrthologID* pipeline. Existing programs are depicted as hexagons (MAFFT, PAUP, and CAOS). Trapezoids indicate *OrthologID* operations implemented by existing software. The rectangles at the bottom represent the product of each step in the pipeline; Step 1: the alignment product, Step 2: the phylogenetic tree product, and Step 3: the generation of diagnostics. Abbreviations: GFG = gene family generator; AC = alignment constructor; TB = tree builder; DG = diagnostics generator.

with a total of 179,005 genes. This database can be continually updated when new fully sequenced and annotated genomes come online. Only sequences from organisms with well-annotated whole genome sequences are used in the construction of guide trees that train the CAOS algorithm to find "diagnostic" amino acid or nucleic acid sites and uses those diagnostics to place an unknown from a less densely sampled genome into an ortholog group. We prefer using genes from only fully sequenced genomes to produce guide trees for two reasons. First, the fully sequenced genomes are also annotated to a better degree than incomplete genomes and EST projects. Second, the absence of a particular gene in a gene family cannot be determined from a partial genome or from EST sequences of the transcriptome of a genome; we consider guide trees constructed from such genomes to be potentially incomplete.

Emanating from the database are four subprograms that perform the following four tasks leading eventually to the construction of a guide tree for a particular gene family.

(1) A gene family generator (GFG; **Fig. 2.1**) that utilizes an e-value based approach to sorting genes into "gene families." Unlike the single linkage cluster algorithm mentioned in **Note 4** or the Coginator that is used to generate COG *(6)* families, we use only raw E-values to sort through the genes to place them in families.

(2) Next an alignment constructor (AC; **Fig. 2.1**) takes the genes placed into a gene family and aligns them with default parameters of the MAFFT *(8)* alignment program.

(3) The gene family alignments are then analyzed in the tree builder (TB; **Fig. 2.1**) using parsimony in PAUP* *(9)*.

(4) The trees are then processed by a diagnostics generator (DG; **Fig. 2.1**) where they are used as guide trees. Diagnostics are extracted from the guide trees using the CAOS algorithm *(3, 10)*.

In this way for each gene family a set of diagnostic rules is generated that can then be used in the Web interface to facilitate the identification of unknown query sequences supplied by the user. Because the diagnostic rules are used to generate the identification, the identifications can also include the diagnostics for inclusion of a query into an ortholog group.

The *OrthologID* program is structured so that other phylogenetic, alignment and DGs can be interchanged. For instance, the basic program can accommodate other alignment programs than MAFFT *(8)*, or other phylogenetic tree building programs than PAUP* *(9)*. In addition, while we prefer parsimony as the method for generating trees, the general program is built to also accommodate likelihood, distance, or Bayesian methods.

To date *OrthologID* has been constructed to facilitate identification of plant orthologs. We are in the process of extending the approach to accommodate other databases, and these new databases will operate the same as the Plant *OrthologID* database. The *OrthologID*website (**Fig. 2.2**) can be accessed at http://nypg.bio.nyu.edu/*orthologID/*. There are three "hot" buttons on this page: an "orthologous group search" button, a "query orthology classification" button, and an "about *OrthologID*" button. These hot buttons gain the user access to the two main ways to interact with *OrthologID*. The first, the "orthologous group search" button, allows the user to access

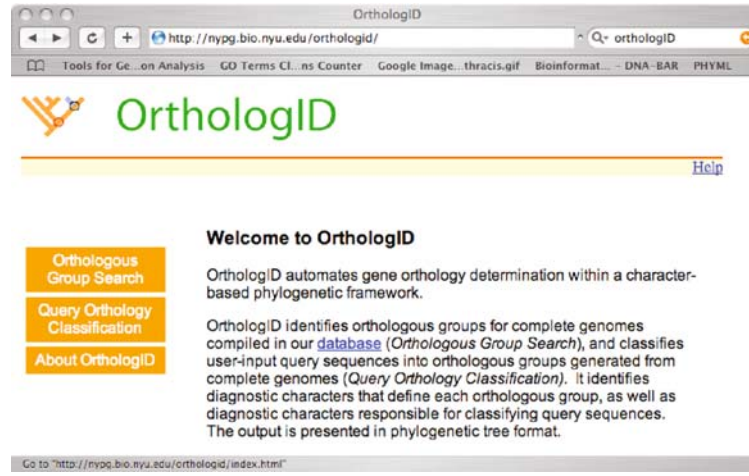


Fig. 2.2. Screenshot of *OrthologID* homepage (http://nypg.bio.nyu.edu/orthologid/).

specific gene family trees and to observe the diagnostics determined by *OrthologID*. This option requires a TAIR gene number for a gene or gene family annotated by The Arabidopsis Information Resource (http://www.arabidopsis.org/) for input. The second way of interacting with *OrthologID* is through the "query orthology classification" button. This option takes the query sequence and attaches it to the gene family tree and shows diagnostics for the whole gene family tree. This option requires a FASTA file of a gene sequence either of known gene family membership or an unknown sequence as input. The third button on the page gains access to the online introduction to *OrthologID*. Below we show worked examples for both options.

3. Examples

3.1. Worked Example # 1. Orthologous Group Search

Press the "orthologous group search" button on the *OrthologID*-homepage. Then, simply paste a query TAIR Gene Family Number into the provided query box (the red circle in **Fig. 2.3**). In this case we have pasted the TAIR number of the malate dehydrogenase gene (AT3G47520) family into the query box.

Press the "Search" button to the right of the query box. If the query is in the Plant *OrthologID* database, the *OrthologID* web server will return the identification of the gene family at the top of the screen (circled in red), a phylogenetic tree on the left and alignment of sequences from gene family members on the right (**Fig. 2.4**). The buttons on the top right of the screen allow the user to scroll across the alignment as indicated.

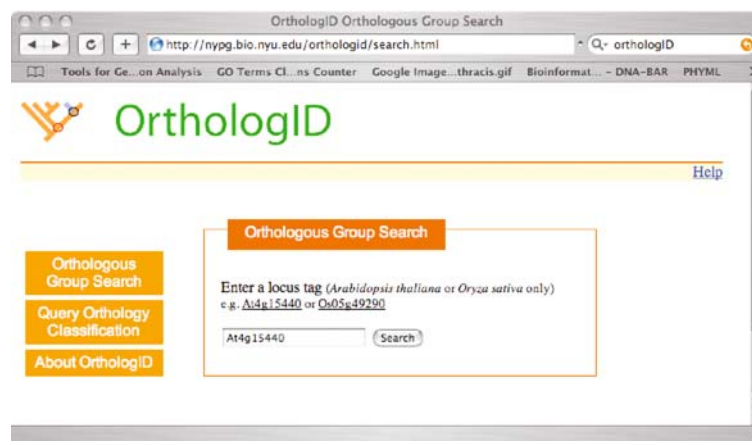


Fig. 2.3. Screenshot of web Group Search in *OrthologID* (http://nypg.bio.nyu.edu/orthologid/search.html).

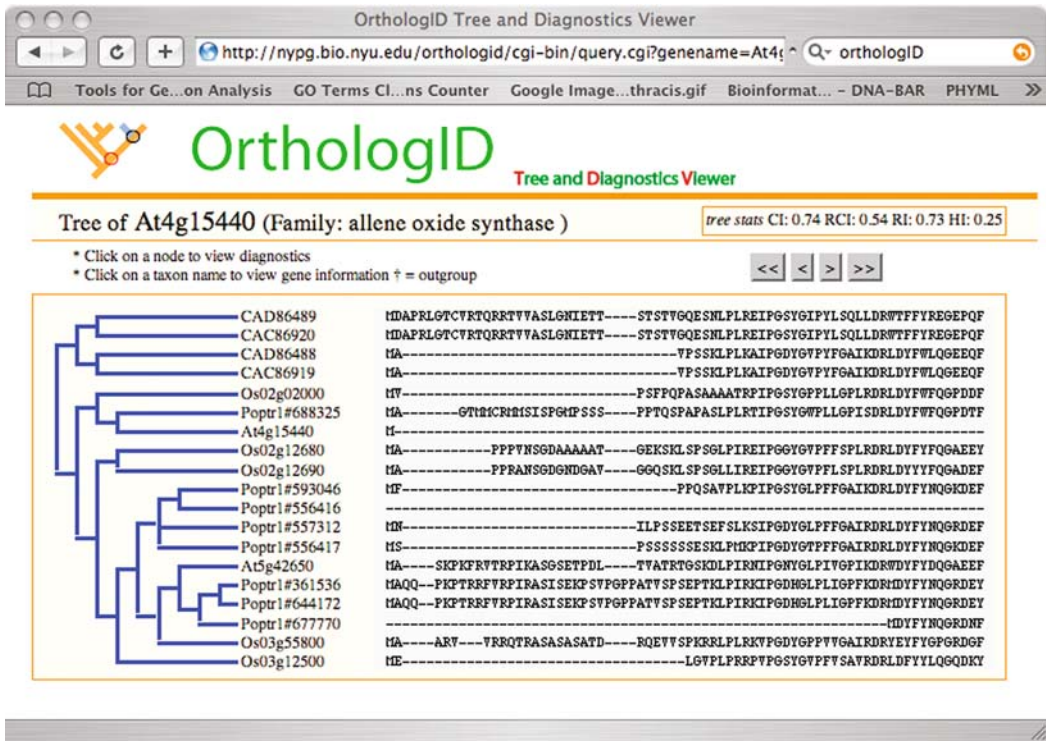


Fig. 2.4. Screenshot of results of querying the Group Search page with AT3G47520. The *arrow* indicates where the diagnostic query for **Fig. 2.5** is clicked.

This screen has two further interactive tools. First, any of the nodes can be clicked on to show the diagnostic amino acid sites and their states for the various ortholog groups displayed (by scrolling over the phylogenetic tree's nodes and clicking on any node). In this example we have clicked on the node near the arrow. When this node is clicked, all of the diagnostic sites in the guide tree are highlighted in red in the accompanying alignment and the group under examination is boxed off in light blue (**Fig. 2.5**).

For any gene in the tree to the right, by simply clicking on the gene number, full information on that gene can be obtained. In this case we have clicked on the *Populus* gene family member 686130 and this links out to the JGI *Populus trichocarpa* website with all of the annotation information for this specific protein.

3.2. Worked Example # 2. Query Orthology Classification

Press the "query orthology classification" button on the *OrthologID*-homepage. Paste a query FASTA sequence from an unknown protein sequence. In this case we have pasted a *Solanum* malate dehydrogenase gene family member into the query box (**Fig. 2.6**). The search button starts the search procedure. If the gene family that the query sequence belongs to is not in the Plant *OrthologID* database, then a "No Hits" response will be returned.

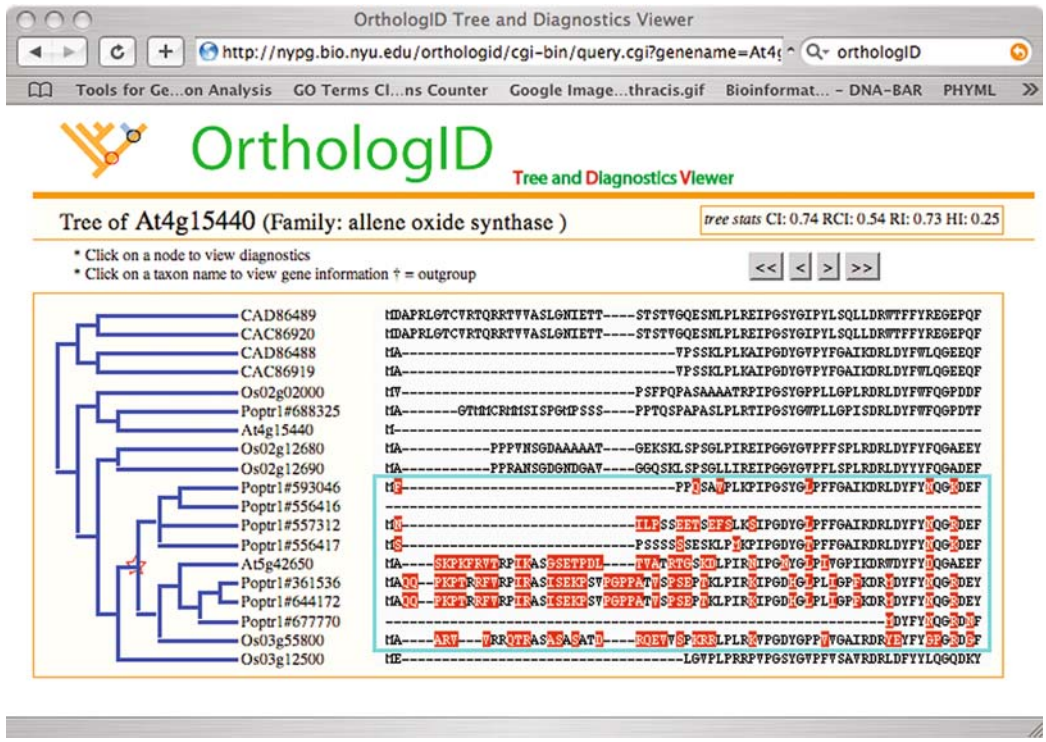


Fig. 2.5. Screenshot of querying the tree at the position marked by an *arrow* in **Fig. 2.4**. A *star* marks the node of interest and the characters that are involved in diagnosis of the group of genes in the box are highlighted.

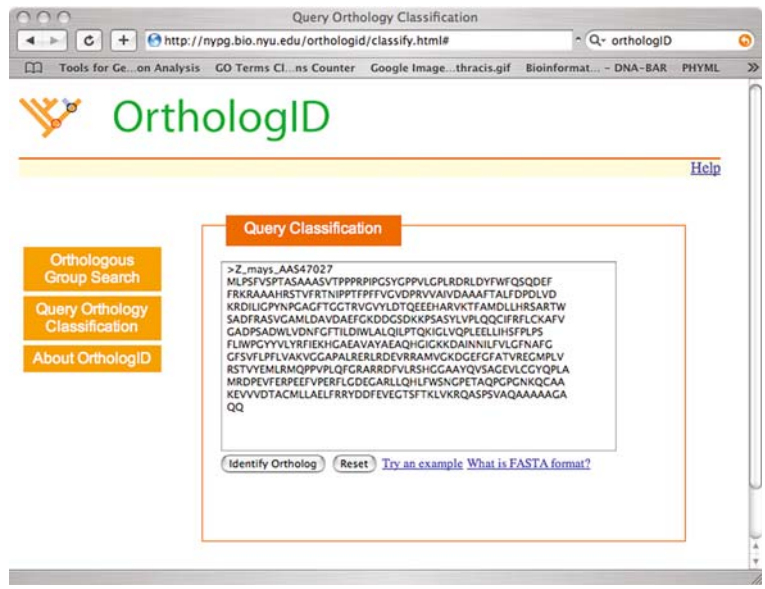


Fig. 2.6. Screenshot of the query ortholog classification in *OrthologID* (http://nypg.bio.nyu.edu/orthologid/classify.html).

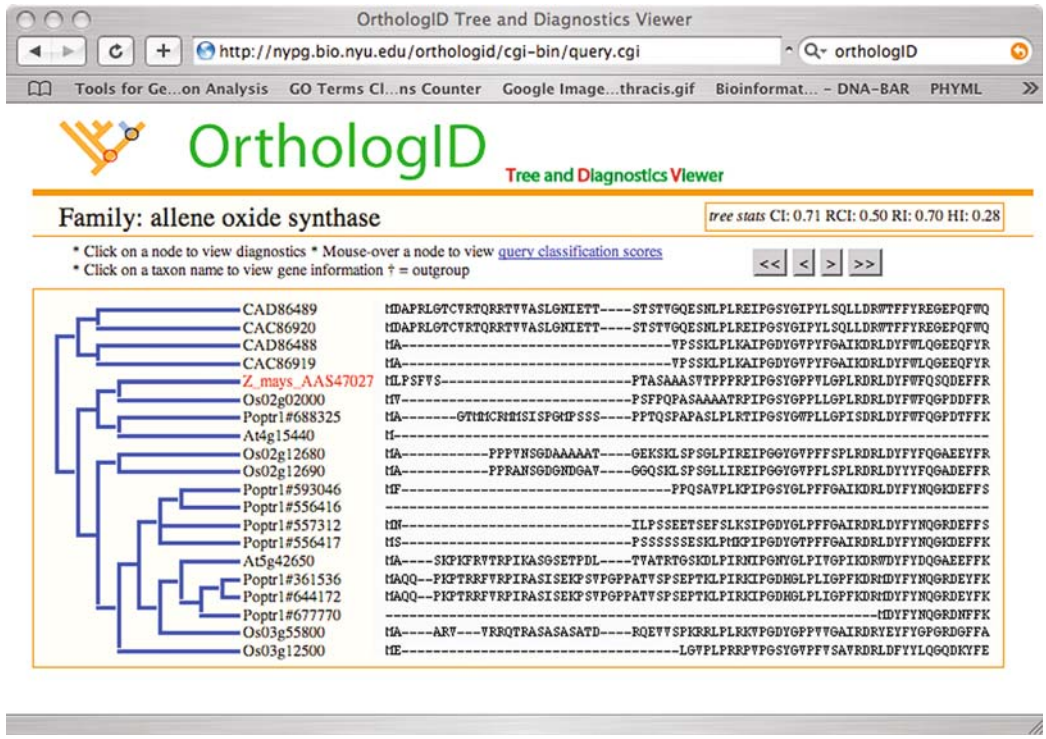


Fig. 2.7. Screenshot of the results of a query sequence placed in the query ortholog classification in *OrthologID*. The *oval* indicates the gene family the query sequence is orthologous to. The query sequence is shown attached to a node in the tree.

If the query sequence home gene family is in the Plant *OrthologID* database, the *OrthologID* web server will return the identification of the gene family at the top of the screen (circled in red), a phylogenetic tree on the left with the query sequence attached to the tree (in red), and an alignment of sequences from gene family members on the right (**Fig. 2.7**).

As with the results from the orthologous group search function, the tree can also return annotated gene information for each gene in the tree, by clicking on the gene name or number, and the diagnostic sites for each of the potential groups in the tree by clicking on the nodes in the tree (**Fig. 2.8**).

Another interactive aspect of the "query orthology classification" page is also shown in **Fig. 2.8**. When the node that contains the original query sequence is clicked, two numbers appear. The first gives the number of characters that are part of the diagnostics and the second number gives a score for the placement of the query sequence into the group it is placed (*see* **Note 6**).

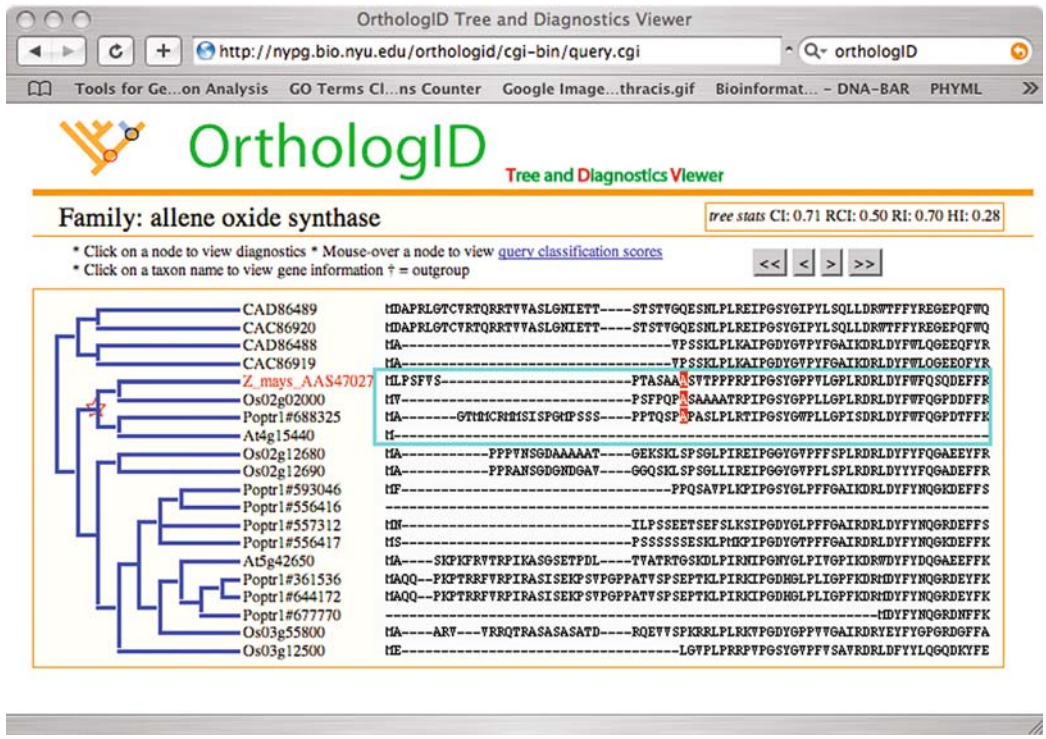


Fig. 2.8. Screenshot of an examination of the placement of the query sequence from **Fig. 2.7**. The box shows the statistics for diagnosing the query sequence to its position in the tree. The diagnostics used to place the query sequence are indicated in the *box* and highlighted.

4. Notes

1. Distances and orthology. Some authors have argued that distance measures are adequate indicators of homology or of significant evolutionary or biological relationships. On the surface this assumption makes some sense. When comparing several protein sequences it is assumed that the two with the greatest similarity are each other's closest relatives and hence, if they originate in different species, they are assumed to be orthologs. Eisen *(11)* and Thornton and DeSalle *(12)* have discussed the ins and outs of distance-based homology studies. They cite four assumptions and problems of this approach: (1) divergence rates are identical in all lineages. This assumption is often not the case in gene family evolution; (2) the pairwise similarity scores include not only phylogenetically informative synapomorphies but also shared ancestral characters (symplesiomorphies) and unique derived ones (autapomorphies); (3) multiple hits problems often violate the additivity of distances; and (4) accurate estimates of divergence are needed.

2. Phylogenetics and orthology assessment via sequence alignment and *OrthologID*. DePinna *(13)* suggested that homology assessment consists of two steps. The first requires that a hypothesis of homology be posed based on some measure of sameness of entities he termed primary homology. This concept should sound familiar and often times this primary homology is established by similarity measures or assessing similarity of two entities, be they genes or organisms. The hypothesis of homology or primary homology statement is then tested using phylogenetic analysis. If the primary homology statement is not rejected, it becomes a secondary homology statement. In this case, the secondary homology statement is a shared derived character or a synapomorphy. Brower and Schawaroch *(4)* refined this homology scheme by adding a step. They begin the assessment of homology with an initial step they call topographic similarity assessment. Once topographic similarity is established, a step involving establishing character state identity follows. The final step is the testing of the hypotheses established in the first two steps. Brower and Schawaroch *(4)* introduced their scheme because they realized a basic difference between molecular and morphological approaches to systematics.
3. The importance of E-values. In addition to information about functionally annotated genes and structure, one of the most commonly used tools for identifying genes is the use of an E-value cutoff. The choice of E-value is somewhat arbitrary though and some studies have noticed a "big genome attraction" effect. This phenomenon results from the idea that organisms with large genomes will include remnants of a large number of gene families. Some researchers have resorted to a process called "conditioning" in order to overcome this problem. In recent attempts to explore "E-value space," researchers examine phylogenies constructed using matrices assembled using a range of E-value cutoffs. Lienau et al. *(14)* examined the impact of E-value choice on the gene presence absence tree of life. When very stringent E-values (very small E-values) are used as a cutoff, the trees become less resolved, but what resolution exists agrees well with what we know about organismal history. The reason for the lack of resolution is that when very small E-values (such as e-300) are used, most of the information about presence/absence of genes is eliminated from a matrix *(14)*.
4. Comparison of *OrthologID* with PhiG. The PhiG approach is a straightforward application of Fitch's *(15)* original solution to the orthology paralogy problem. The approach involves five steps: "(1) an all against all BLASTP of the complete proteomes; (2) global alignment and distance calculation of the gene pairs identified by BLAST; (3) iterative, hierarchical clustering; (4) multiple sequence alignment (MSA) creation

and editing; and (5) gene tree reconstruction" *(7)*. Note that these steps are also very similar to the steps outlined in **Table 2.1**, with steps 1, 2, and 3 approximating the topographical similarity step in the Gfh approach, step 4 coinciding with character state assessment, and step 5 coinciding with the last step of all three approaches discussed in **Table 2.1**. The PhiG website adds an interesting aspect to gene family identification by putting the orthology statements derived from their five-step procedure into a chromosomal location context. In addition to this aspect of PhiG that relates to fully sequenced genomes, the package also uses a Hidden Markov Model (HMM) approach to facilitate the classification of putative proteins from less densely sampled genomes such as those generated by the EST approach.

5. The CAOS algorithm *(3, 10)*. The CAOS approach is based on population aggregation analysis as articulated by Davis and Nixon *(16)* and is used to discover the diagnostics in the *OrthologID* approach. In essence the guide tree allows for a phylogenetic grouping of protein sequences that can be used by the CAOS algorithm to find diagnostics. Diagnostics can be "single pure," "compound private pure," and even "compound nonprivate (polymorphic) pure" (see OrthologID website – http://nypg.bio.nyu.edu/orthologid/diagnostics.html). The compound classes of diagnostics take advantage of combining columns that in and of themselves are not diagnostic. The compound private pure diagnostics take advantage of private character states in particular ortholog groups and the compound polymorphic pure diagnostics utilize complex combinations of character states in positions to discover diagnostics in information that at first seems simply polymorphic.
6. Interpreting *OrthologID* scores. **Figure 2.9** shows an example of how to interpret the two numbers that are given when scrolling over the query sequence node on the "query orthology classification" page. In **Fig. 2.9**, there are the two examples we used to explain these numbers. In both, the first number indicates the number of diagnostic characters shared between the query and the gene(s) belonging to the clade it is placed into. In both the examples shown, this number is 50. Note that the absolute number of diagnostic characters may not be a reliable indication of how decisively the query is being placed into a clade. The second number, represented as a percentage score, is a better indication of the strength of query classification. In example 1, query sequence *Q1* shares 50 characters ($a = 50$) with the gene(s) in clade A, and zero characters ($b = 0$) with the gene(s) in clade B. As a result, *OrthologID*places *Q1* into clade A with a percentage score of

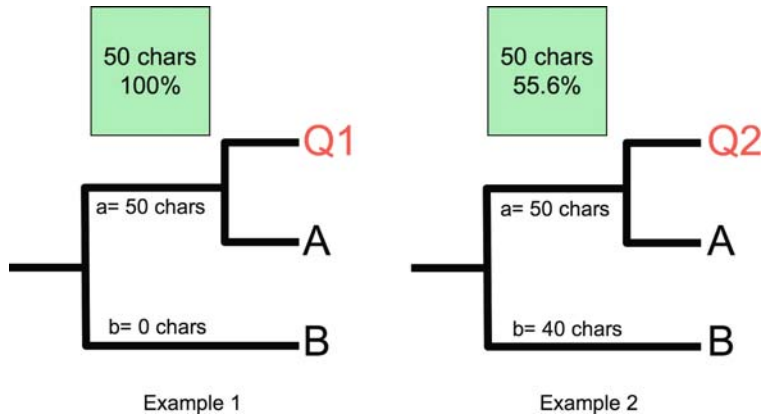


Fig. 2.9. An example of how to interpret the two numbers that are given when scrolling over the query sequence node on the "query orthology classification" page. The first number in both examples indicates the number of diagnostic characters shared between the query and the gene(s) belonging to the clade it is placed into. The second number, represented as a percentage score, is the better indication of the strength of query classification. In example 1, query sequence *Q1* shares 50 characters ($a = 50$) with the gene(s) in clade A, and zero characters ($b = 0$) with the gene(s) in clade B. As a result, *OrthologID* places *Q1* into clade A with a percentage score of 100%. The percentage score is calculated using the formula $(a/(a+b)) \times 100\%$. In example 2, $a = 50$ and $b = 40$; as a result, the percentage score is lower (55.6%), indicating that the strength of query placement for *Q2* is weaker than that of *Q1* in example 1.

100%. The percentage score is calculated using the formula $(a/(a+b)) \times 100\%$. In example 2, $a = 50$ and $b = 40$; as a result, the percentage score is lower (55.6%), indicating that the strength of query placement for *Q2* is weaker than that of *Q1* in example 1.

References

1. Chiu, J. C., Lee, E. K., Egan, M. G., Sarkar, I. N., Coruzzi, G. M., and DeSalle, R. (2006) *OrthologID*: automation of genome-scale ortholog identification within a parsimony framework. *Bioinformatics* **22**, 699–707.
2. Koski, L. B., and Golding, G. B. (2001) The closest BLAST hit is often not the nearest neighbor. *J Mol Evol* **52**, 540–42.
3. Sarkar, I. N., Thornton, J. W., Planet, P. J., Figurski, D. H., Schierwater, B., and DeSalle, R. (2002) An automated phylogenetic key for classifying homeoboxes. *Mol Phylogenet Evol* **24**, 388–99.
4. Brower, A. V. Z., and Schawaroch, V. (1996) Three steps of homology assessment. *Cladistics* **12**, 265–72.
5. Altschul, S. F., Gish, W., Miller, W., Meyers, E. W., and Lipman, D. J. (1990) Basic local alignment search tool. *J Mol Biol* **215**, 403–10.
6. Tatusov, R. L., Galperin, M. Y., Natale, D. A., and Koonin, E. V. (2000) The COG database: a tool for genome-scale analysis of protein functions and evolution. *Nucleic Acids Res* **28**, 33–6.
7. Dehal, P. S., and Boore, J. L. (2006) A phylogenomic gene cluster resource: the Phylogenetically Inferred Groups (PhIGs) Database. *BMC Bioinformatics* **7**, 201.
8. Katoh, K., Kuma, K., Toh, H., and Miyata, T. (2005) MAFFT version 5: improvement in accuracy of multiple sequence alignment. *Nucleic Acids Res* **33**, 511–18.

9. Swofford, D. L. (2003) *PAUP* Phylogenetic Analysis Using Parsimony (*and Other Methods)*. Version 4. Sinauer Associates, Sunderland, Massachusetts.
10. Sarkar, I. N., Planet, P. J., Bael, T. E., Stanley, S. E., Siddall, M., DeSalle, R., and Figurski, D. H. (2002) Characteristic attributes in cancer microarrays. *J Biomed Inform* Apr/May; **35**(2), 111–22
11. Eisen, J. A. (1998) Phylogenomics: improving functional predictions for uncharacterized genes by evolutionary analysis. *Genome Res* **8**, 163–67.
12. Thornton, J. W., and DeSalle, R. (2000) Phylogenetics meets genomics: homology and evolution in gene families. *Annu Rev Genomics Hum Gene* **1**, 43–72.
13. DePinna, M. C. C. (1991) Concepts and tests of homology in the cladistic paradigm. *Cladistics* **7**, 367–94.
14. Lienau, E. K., DeSalle, R., Rosenfeld, J. A., and Planet, P. J. (2006) Reciprocal illumination in the gene content tree of life. *Syst Biol* **55**, 441–53.
15. Fitch, W. M. (1970) Distinguishing homologous from analogous proteins. *Syst Zool* **19**, 99–113.
16. Davis, J. J., and Nixon, K.C. (1992) Populations, genetic variation and the delimitation of phylogenetic species. *Syst Biol* **41**, 121–35.

Chapter 3

Multiple Alignment of DNA Sequences with MAFFT

Kazutaka Katoh, George Asimenos, and Hiroyuki Toh

Abstract

Multiple alignment of DNA sequences is an important step in various molecular biological analyses. As a large amount of sequence data is becoming available through genome and other large-scale sequencing projects, scalability, as well as accuracy, is currently required for a multiple sequence alignment (MSA) program. In this chapter, we outline the algorithms of an MSA program MAFFT and provide practical advice, focusing on several typical situations a biologist sometimes faces. For genome alignment, which is beyond the scope of MAFFT, we introduce two tools: TBA and MAUVE.

Key words: Multiple sequence alignment, progressive method, iterative refinement method, consistency objective function, genome comparison.

1. Introduction

Multiple sequence alignment (MSA) is used in various phases of biological studies, such as in SSU rRNA-based phylogeny inference for organismal classification *(1)*, or in informant sequence analysis for comparative gene finding *(2)*. Most current MSA programs successfully create an accurate alignment in a reasonable amount of time for easy cases, in which the number of input sequences is small (several dozen), the length of each sequence is short (up to ~500 nucleotides), and the sequences are highly similar (percent identity of ~70) to each other. Such cases require no special consideration. If all of these conditions are not met, however, then choosing appropriate tools and configurations, while considering the tradeoff between accuracy and computational cost, can be difficult.

Here we explain the basic algorithms implemented in an MSA program, MAFFT *(3, 4)*, and then explain how to select an appropriate option. Recent benchmark studies *(5, 6)* on RNA and DNA consistently identified MAFFT as one of the most accurate MSA

David Posada (ed.), *Bioinformatics for DNA Sequence Analysis,* Methods in Molecular Biology 537

DOI 10.1007/978-1-59745-251-9_3

methods in a wide range of situations. However, MAFFT cannot process all of the DNA alignment problems we may encounter. For example, MAFFT assumes that the order of aligned sites or blocks is completely conserved among all of the sequences. This assumption can be violated by genome inversions, translocations, or duplications. That is, genome alignment is beyond the scope of MAFFT. Thus we also explain two tools specialized to genome alignment: TBA *(7, 8)* and MAUVE *(9, 10)*. In addition, it is important to try out multiple different methods in order to obtain a high-quality alignment *(6, 11)*.

1.1. Basic Concepts

A sequence alignment is a set of corresponding residues among a collection of nucleotide or amino acid sequences. Aligned residues are usually interpreted to share an evolutionary origin and/or to be functionally equivalent. We mention here the alignment of multiple (>2) DNA sequences. A pairwise sequence alignment (composed of two sequences) can be calculated by the dynamic programming (DP) algorithm *(12–14)*. An MSA is usually displayed as a table, in which corresponding residues occupy the same column, as in **Fig. 3.1c**. When a sequence has no corresponding residue because of an insertion or deletion event, the position is displayed as "-" which is called a "gap." The sequences can be protein- or RNA-coding sequences or non-coding nucleotide sequences, such as introns or spacers.

The sequences involved in an MSA are usually assumed to be homologous, that is, derived from a single common ancestral sequence. Some MSA methods assume global homology while some other methods assume only a local homology (*see* **Note 2**). Most MSA programs do not attempt to filter non-homologous sequences, leaving the decision of what sequences to include in the MSA as an external decision for the user. However, this problem is sometimes important in comparative analysis, as discussed in **Section 3.3**.

1.2. MSA Strategies in MAFFT

1.2.1. Progressive Method

The progressive method *(15)* is currently the most widely used multiple alignment algorithm. ClustalW *(16)* and most of the other packages use this strategy with various modifications. In this strategy, a guide tree is created based on all-to-all pairwise comparisons, and an MSA is constructed using a group-to-group alignment algorithm at each node of the guide tree. In principle, the DP algorithm for pairwise alignment can be extended to that for group-to-group alignment. To achieve a reasonable balance between speed and accuracy, MAFFT uses a two-cycle progressive method (FFT-NS-2) shown in **Fig. 3.2**. In this method, low-quality all-pairwise distances are rapidly calculated, a tentative MSA is constructed, refined distances are calculated from the MSA, and then the second progressive alignment is performed (*see* **Note 1**). FFT-NS-1, a one-cycle progressive method, is faster and less accurate than FFT-NS-2. There is a more scalable option, PartTree *(17)*, applicable to ~50,000 sequences.

a

```
>Sequence1
nnnnnnattccggttgatcctgccggaggccactgctatcggggtccgactaagccatgc
gagtcaaggggctcccttcggggagcaccggcgcacggctcagtaacacgtggctaacct
accctcgggtgggggataacctcgggaaactgaggctaatcccccatagggaggaggtc
tggaatgatccctcccgaaaggcgtaagctgcccgaggatggggctgcggcggattagg
agtgaggccctgtccgcaagggcagggtcgaacttgggttcagcgaggggggcgaagtcg
taacaaggtagccgtaggggaactgcggctggatcacctccnnnnn
>Sequence2
nnnnnnactccggttgatcctgccggaggccaccgctatcggggtccgactaagccatgc
aagtcgagggccgcccggcaatgggggcggcccggcggacggctcagtaacacgtgggta
acctaccctcgggacggggataaccccggcgaaagtggggctaatccccgataggcgggg
cggcctggaacggtcctccgccgaaagggcccgggcccatgccgcccgggtccgcccgag
gatgggcctgcggccgattaggtagttggcggggtaacggcccgccaagccgataatcgg
tacgggcggtgagagccggagcccggagacggggactga
>Sequence3
agtggggaatcttggacaatgggggaaaccctgatccagcgacgccgcgtgcgggacgaa
gcccttcggggtgtaaaccgctgtggcgggggaagaataaggtagggaggaaatgcccta
tgtcggatgtgaaatcccacggctcaaccgtggggctgcatccgaaactaccaggcttgg
tct
```

b

```
>Sequence1
nnnnnnattccggttgatcctgccggaggccactgctatcggggtccgactaagccatgc
gagtcaaggggctccc----ttcggggagcaccggcgcacggctcagtaacacgtggcta
acctaccctcgggtgggggataacctcgg-gaaactgaggctaatcccccatagggaggg
aggtctggaatgatccctcccgaaagg----------------cgtaagctgcccgag
gatggggctgcggcggattaggagtgaggccctgtccgcaagggcagggtcgaacttggg
ttcagcgaggggggcgaagtcgtaacaaggtagccgtag---gggaactgcggctggatc
acctccnnnnn
>Sequence2
nnnnnnactccggttgatcctgccggaggccaccgctatcggggtccgactaagccatgc
aagtcgagggccgcccggcaatgggggcggcccggcggacggctcagtaacacgtgggta
acctaccctcgggacggggataaccccggcgaaagtggggctaatccccgataggcgggg
cggcctggaacggtcctccgccgaaagggcccgggcccatgccgcccgggtccgcccgag
gatgggcctgcggccgattaggtagttggcggggt-----aacggcccgccaagccgata
atcggtacgggcggtgag-------------agccggagcccggagacggggactga---
-----------
>Sequence3
ag--------tggggaatcttg--gacaatgggggaaaccctgatccagcgacgccgcgt
gcgggacgaagccctt----cggggtgtaaaccgctgtg---------------------
--------gcgggggaagaataaggtagg--------gaggaaatgccctatgtcggatg
tgaaatcccacg-----------------------------------------gctcaac
cgtggggctgcatccga--aactaccaggcttggtct----------------------
------------------------------------------------------------
-----------
```

c

```
CLUSTAL W (1.83) multiple sequence alignment

Sequence1       --NNNNNNATTCCGGTTGATCCTGCCGGAGGCCACTGCTATCGGGGTCCGACTAAGCCAT
Sequence2       --NNNNNNACTCCGGTTGATCCTGCCGGAGGCCACCGCTATCGGGGTCCGACTAAGCCAT
Sequence3       AGTGGGGAATCTTGGACAAT------GGGGGAAACCCTGATCCAG---CGACGCCGC-GT
                         *    **   **       ** **  **    ***  *   ****   **  *

Sequence1       GCGAGTCAAGGGGCTCCCTTC----GGGGAGCACCGGCGCACGGCTCAGTAACACGTGGC
Sequence2       GCAAGTCGAGGGCCGCCCGGCAATGGGGGCGGCCCGGCGGACGGCTCAGTAACACGTGGG
Sequence3       GCGGGACGAAGCCCTTCGGGGTGT-AAACCGCTGTGGCGGG-GGAAGAATAAGGTAGGGA
                **  * * * *  *  *             *    ****   **   * ***     **

Sequence1       TAACCTACCCT---CGGGTGGGGGATAACCTCGG-GAAACTGAGGCTAATCCCCCATAGG
Sequence2       TAACCTACCCT---CGGGACGGGGATAACCCCGGCGAAAGTGGGGCTAATCCCCGATAGG
Sequence3       GGAAATGCCCTATGTCGGATGTGAAATCCCACGGCTCAACCGTGGGG------CTGCATC
                  *  * ****     **  * * *   ** ***   **  * **        *   *

Sequence1       GGAGGAGGTCTGGAATGATCCCTCCCCGAAAGGCGTAAGCTGCCCGAGGATGGGGCTGCG
Sequence2       CGGGGCGGCCTGGAACGGTCCTCCGCCGAAAGGGCCCGG--GCCCATG---CCGCCCGGG
Sequence3       CGAAACTACCAGGCTTGGTCT-----------------------------------------
                 *       * **   * **

Sequence1       GCGGATTAGGAGTGAGGCCCTGTCCGCAAGGGCAGGGTCGAACTTGGGTTCAGCGAGGGG
Sequence2       TCCGCCCGAGGATGGGCCTGCGGCCGATTAGGTA--GTTGG---CGGGGTAA---CGGCC
Sequence3       ------------------------------------------------------------


Sequence1       GGCGAAGTCGTAACAAGGTAGCCGTAG-GGGAACTGCGGCTGGATCACCTCCNNNNN
Sequence2       CGCCAAGCCGATAATCGGTACGGGCGGTGAGAGCCGGAGCCCGGAGACGGGGACTGA
Sequence3       ------------------------------------------------------------
```

Fig. 3.1. (**a**) Multiple unaligned sequences in the FASTA format. (**b**) Multiple alignment in the FASTA format. (**c**) Multiple alignment in the CLUSTAL format.

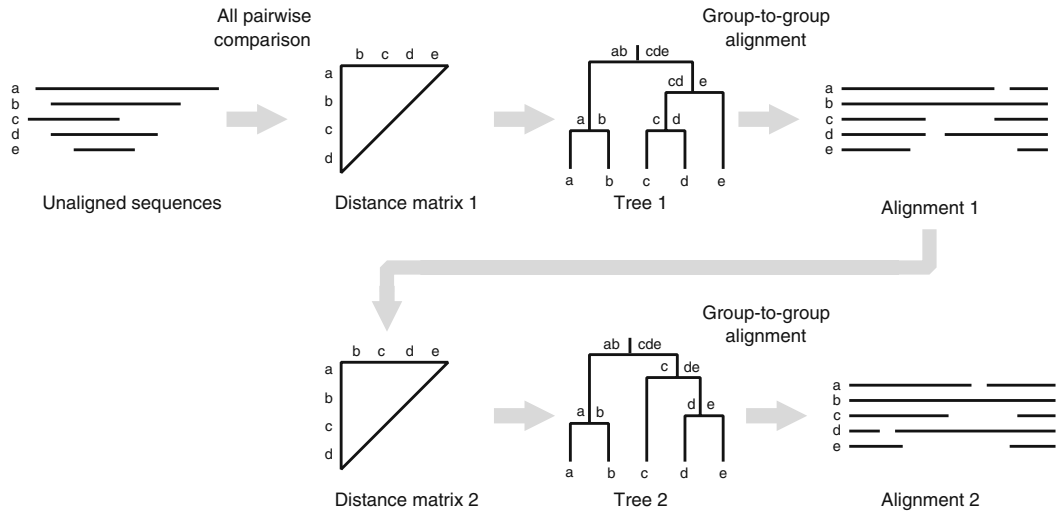


Fig. 3.2. Calculation procedure of the progressive method.

The progressive method has a drawback in that once a gap is incorrectly introduced, especially at an early step (near a leaf of the guide tree), the gap is never removed in later steps. To overcome this drawback, there are two types of solutions: the iterative refinement method and the consistency-based method. These two procedures are quite different; the former tries to correct mistakes in the initial alignment, whereas the latter tries to avoid mistakes in advance. However, both work well to improve alignment accuracy.

1.2.2. Iterative Refinement Method

In the iterative refinement method, an objective function that represents the "goodness" of the MSA is explicitly defined. An initial MSA, calculated by the progressive or another method, is subjected to an iterative process and is gradually modified so that the objective function is maximized, as shown in **Fig. 3.3**. Various

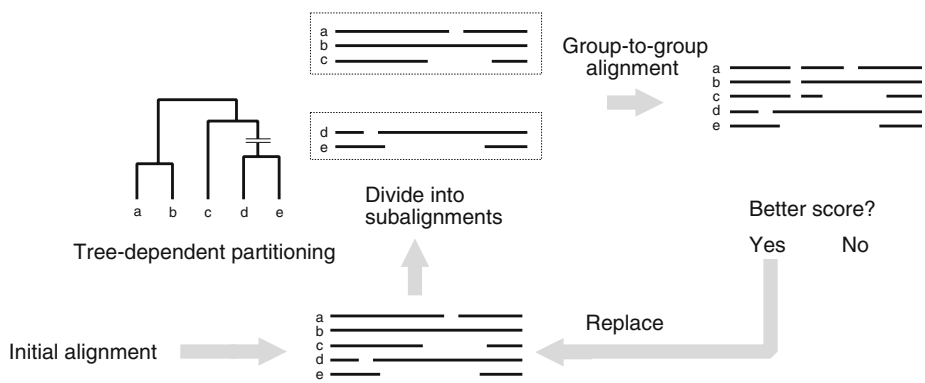


Fig. 3.3. Calculation procedure of the iterative refinement method.

combinations of objective functions and optimization strategies have been proposed to date *(18–22)*. Among them, Gotoh's iterative refinement method, PRRN with the weighted sum-of-pairs (WSP) objective function *(23–25)*, is the most successful one, and it forms the basis of recent iterative refinement methods including the FFT-NS-i option of MAFFT. In the iterative refinement method, an MSA is partitioned into two groups, and the two groups are re-aligned using a group-to-group alignment algorithm. This process is repeated until no more improvements are made. To save computation time, the partitions of the MSA are restricted to those corresponding to the branches of the guide tree *(26)* (*see* **Note 1**).

1.2.3. Consistency-Based Iterative Refinement Method

Consistency-based method is an entirely different approach to overcome the drawback of the progressive method. In consistency criteria, the accuracy of an MSA is judged with regard to how it is consistent with pairwise alignments. Several types of consistency criteria were described previously *(27–29)*, but TCoffee *(30)* achieved a great improvement in accuracy (*see* **Note 3**). MAFFT adopted a consistency criterion similar to COFFEE *(29)* and combined it with the WSP objective function. The combined objective score is iteratively maximized in the G-INS-i, L-INS-i, and E-INS-i options *(4)*. The three options should be separately used according to whether the homology is expected to be global or local (*see* **Note 2** and **Section 2.3**).

1.3. Tools for Genome Alignment

Along with the availability of genome sequences has come the need to construct genomic alignments. Since genomes undergo various evolutionary events such as inversions, translocations, and duplications, the original order and orientation of the nucleotides is no longer maintained. This makes the alignment problem harder, because genome alignment tools need to detect the homologous pieces as well as perform a nucleotide alignment of them, instead of performing one global alignment of the whole input. Since some homologous regions might not appear in all input genomes, output alignments may relate a subset of the input sequences.

Determining the homologous pieces is complicated by the presence of paralogs and segmental duplications, which lead to an increase in the number of possible combinations when many genomes are involved. There are different paradigms followed by the alignment tools in order to handle duplicated bases. **Figure 3.4** (*i*) illustrates a toy example with three genomes (A, B, and C, depicted horizontally) and two genes (1 and 2). Genome A contains two copies of gene 1 (A1 and A1′, both orthologous to B1 and C1), and genome B contains two copies

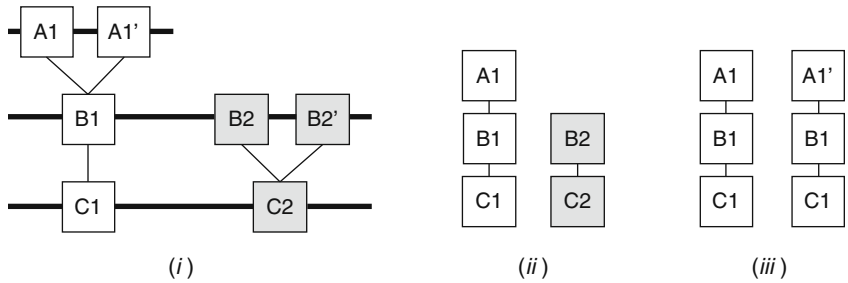


Fig. 3.4. An example of three genomic sequences with tamdemly duplicated genes.

of gene 2 (B2 and B2′, both orthologous to C2). The ideal output would include an alignment containing A1, A1′, B1, and C1, and another alignment containing B2, B2′, and C2. Traditional multiple alignment output does not allow a given species to appear more than once in the same alignment; for example, B2 and B2′ cannot appear in the same alignment. The relationship between B2, B2′, and C2 could be represented by two separate alignments (one between B2 and C2, and another between B2′ and C2), in which case the sequence of C2 would appear twice in the output, in two different alignments. What happens in practice is that most multiple alignment tools allow each base from each genome to participate in at most one alignment, and therefore provide a one-to-one mapping between the bases. An example of such an output is shown in **Fig. 3.4** (*ii*), where each gene can only appear once, and two genes from the same genome cannot appear in the same alignment; thus, one of the two copies of each duplicated gene (in this case A1′ and B2′) remains unaligned.

Other alignment tools perform a multiple alignment with respect to a reference genome: the bases of the reference genome must appear exactly once, but the bases of the other genomes are allowed to appear in more than one alignment. Alignments that do not contain the reference genome are not constructed. **Figure 3.4** (*iii*) shows an example of a multiple alignment which uses A as a reference genome; there is exactly one alignment for each of the A1 and A1′ genes and they both contain B1 and C1. In that case, gene 2 is not aligned since it is not present in A. An alignment with respect to a reference genome *G* is suitable for studies that need to address *G*-centric questions, such as "How conserved is a given base of genome *G* " or "How much of genome *G* is covered by each species," etc.

Performing a nucleotide alignment at the whole-genome level can be a resource-intensive process; the analyses that can be performed on a single desktop computer are limited by the size of the sequences involved. The alignment problem is easier for prokaryotes, which have relatively short genomes and are

less repetitive, and is more complex for eukaryotes as the input is longer and more repetitive. Most modern programs can handle (on a single computer) up to ~100 genomes that have up to ~10 M bases each, though the actual complexity depends on the similarity of the genomes and is hard to estimate. Very large alignment problems require either parallelization on a computer cluster, or some other non-trivial setup; however, for any available large genome, an alignment between that genome and related genomes may already exist in some Internet resource.

In **Section 2.6**, we explain how to use two genome alignment tools: TBA and MAUVE. Other alignment tools that are simply too slow for modern alignment problems, that are able to provide genomic alignments for only two genomes (such as BLAST and all its variants), or that require the sequences to be free from rearrangements are not covered here.

2. Program Usage

MAFFT accepts a set of unaligned sequences in FASTA format (**Fig. 3.1a**) and returns an alignment of the sequences in FASTA (**Fig. 3.1b**) or CLUSTAL (**Fig. 3.1c**) format. The following methods are implemented: progressive method (FFT-NS-1 and FFT-NS-2), iterative refinement method (FFT-NS-i), and consistency-based iterative refinement method (G-INS-i, L-INS-i, and E-INS-i). MAFFT is a command-line program with no graphical user interface. It runs on a UNIX-like environment, such as Linux, Terminal on Mac OS X, and Cygwin on Windows. For users who prefer a graphical interface, web-based interfaces are available at several MAFFT servers (**Table 3.1**), but we do not explain here how to use them.

Table 3.1
MAFFT servers

Kyushu University	http://align.bmr.kyushu-u.ac.jp/mafft/online/server/
EBI	http://www.ebi.ac.uk/mafft/
GenomeNet	http://align.genome.jp/mafft/
Max Planck Institute	http://toolkit.tuebingen.mpg.de/mafft
MyHits, SiB	http://myhits.isb- sib.ch/cgi-bin/mafft

MAFFT accepts the following command line arguments that control MSA strategy (See the MAFFT page for the full list of arguments).

- --**retree** ***n*** Progressive alignment is repeated ***n*** times renewing the guide tree. When $n = 0$, only a guide tree is calculated and no alignment is computed. $n = 2$ by default.
- --**maxiterate** ***n*** The number of cycles of iterative refinement. $n = 0$ by default.
- --**6merpair** Distance between every pair of sequences is calculated based on the number of shared 6mers *(3, 31, 32)*. Enabled by default.
- --**globalpair** All pairwise alignments are computed with the Needleman–Wunsch *(12)* algorithm. Disabled by default.
- --**localpair** All pairwise alignments are computed with the Smith–Waterman *(13)* algorithm. Disabled by default.
- --**genafpair** All pairwise alignments are computed with a local alignment algorithm with the generalized affine gap cost *(33)*. Disabled by default.

Various MSA strategies can be executed by combining the above arguments. Users have to select appropriate strategies according to the nature of input sequences, considering the trade-off between accuracy and computational cost.

As the easiest way, there is a fully automated option that selects an appropriate strategy from L-INS-i, FFT-NS-i, and FFT-NS-2, depending on the size of the data.

```
% mafft --auto input_file > output_file
```

However, we strongly recommend seeing how each strategy works and trying some different ones, particularly when a set of distantly related sequences is being aligned. We list several typical cases with possible difficulties and show what option of MAFFT can be helpful. We also show command-line arguments.

2.1. A Large Number (> ~1000) of Sequences – FFT-NS-2, FFT-NS-1, or PartTree

Only the progressive method can align many (> ~1000) sequences in a reasonable amount of time. As the distance calculation process is the time-limiting factor in this case, progressive methods with fast distance calculation are suitable. An approximate distance between a pair of sequences can be roughly estimated by the number of shared k-mers between them *(3, 31, 32)*.

```
% mafft input_file > output_file
```

This command performs the two-cycle progressive method (FFT-NS-2) shown in **Fig. 3.2**. For faster computation, try the one-cycle progressive method (FFT-NS-1)

```
% mafft --retree 1 input_file > output_file
```

which omits the re-estimation of tree, and thus is approximately two times faster than the two-cycle progressive method.

When handling larger numbers of sequences (N>∼10,000), FFT-NS-1 can be still too slow. For that case, MAFFT has a faster and more scalable option, PartTree *(17)*, which relies on an approximate guide tree construction method with a time complexity of $O(N \log N)$. For two-cycle progressive alignment,

```
% mafft --parttree input_file > output_file
```

and for one-cycle progressive alignment

```
% mafft --parttree --retree 1 input_file > output_file
```

Note that these methods are less accurate by 2 or 3% than the corresponding methods with a full guide tree.

2.2. Long Sequences (> ∼10,000 nt.) – FFT-NS-2 or FFT-NS-i

When handling long sequences, space complexity should be considered. FFT-NS-2 and FFT-NS-i automatically switch the DP algorithm to a memory saving one *(34)* requiring only an $O(L)$ RAM space when the length of the alignment exceeds a threshold, where L is sequence length. The FFT-based alignment algorithm *(3)* plays an important role to reduce the calculation time in cases of highly conserved and long sequences. This algorithm is also enabled by default. Thus no special argument is needed. For the progressive method,

```
% mafft input_file > output_file
```

and for the iterative refinement method,

```
% mafft --maxiterate 10 input_file > output_file
```

The maximum number of the iterative refinement cycle is limited to 10 in this command.

2.3. Low Similarity – E-INS-i, L-INS-i, or G-INS-i

First of all, check whether the set of sequences is globally alignable or locally alignable: will the resulting alignment be like that in **Fig. 3.5a** (needs long internal and terminal gaps), **b** (needs long terminal gaps), or **c** (globally alignable)? In general, we recommend

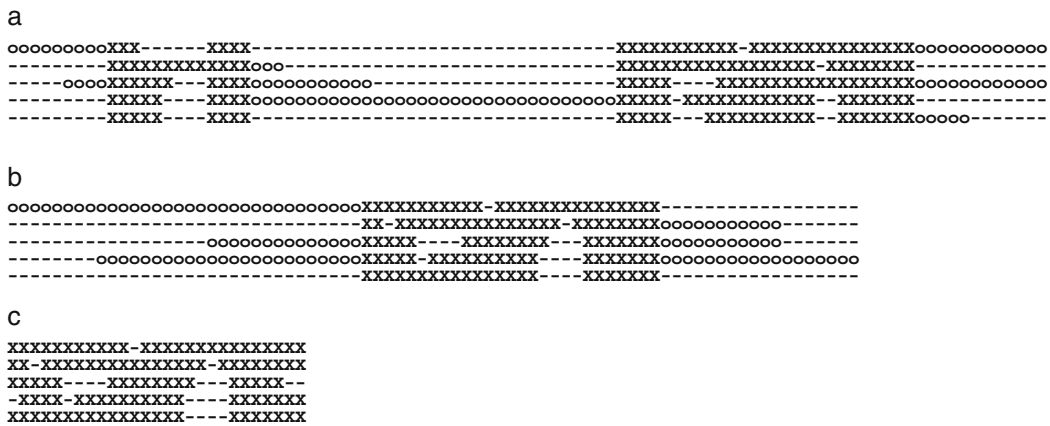


Fig. 3.5. Globally alignable, locally alignable, or long internal gaps? "-" represents a gap, "X" represents an aligned nucleotide and "o" is an un-alignable nucleotide.

assuming case **a** if the type of alignment is unclear, because the assumption of case **a** is the safest of the three. Once an initial alignment is obtained, then trimming the alignment to include only the relevant homologous parts can be done manually, and then the methods designed for case **c** can be applied. In addition, consider the tradeoff between computational costs and accuracy: highly accurate methods tend to be time- and space-consuming. The three methods (G-INS-i, L-INS-i, and E-INS-i) explained here can process only $< \sim 500$ sequences on a standard desktop computer.

Case a – E-INS-i When long internal gaps are expected. This case corresponds to an SSU rRNA alignment composed of distantly related organisms, such as from the three different domains (Eukarya, Bacteria, and Archaea), or a cDNA alignment with splicing variants, both of which need long gaps to be inserted. E-INS-i is suitable for such data. It employs a generalized affine gap cost *(33)* in the pairwise alignment stage, in which the un-alignable regions are left unaligned.

```
% mafft --genafpair --maxiterate 1000 input_file >
  output_file
```

An alias is available:

```
% mafft-einsi input_file > output_file
```

Case b – L-INS-i Locally alignable. When only one alignable block surrounded by long terminal gaps is expected, L-INS-i is recommended.

```
% mafft --localpair --maxiterate 1000 input_file >
  output_file
```

or

```
% mafft-linsi input_file> output_file
```

Case c – G-INS-i Globally alignable. If the entire region of input sequences is expected to be aligned, we recommend G-INS-i that assumes global homology.

```
% mafft --globalpair --maxiterate 1000 input_file>
  output_file
```

or

```
% mafft-ginsi input_file> output_file
```

To obtain a high-quality MSA from the biological point of view, we recommend trying multiple independent methods (*see* **Note 5**), different parameter sets (*see* **Note 4**), and comparing various alignments by eye (*see* **Note 6**).

2.4. Adding New Sequence(s) to an Existing Alignment

In this section, we explain how to align a group of aligned sequences with another group of aligned sequences or with unaligned sequences. The tools explained here can be useful for a semi-automatic alignment or for combining "eye-ball" alignments and automated alignments, although they are not needed for fully automated sequence analyses.

2.4.1. Profile–Profile Alignment

An alignment between two alignments or between an alignment and a single sequence is basically performed by converting each alignment/sequence to a profile *(35)* and then applying a pairwise profile–profile alignment algorithm. Many packages including MAFFT offer this utility.

```
% mafft-profile aligned_group1 aligned_group2> output_file
```

where ***aligned_group1*** and ***aligned_group2*** are alignment files in the FASTA format. Each of them is converted to a profile, and then an alignment between ***aligned_group1*** and ***aligned_group2*** is calculated. Note that this method assumes that the phylogenetic relationship is like that in **Fig. 3.6a** or **3.6b**.

2.4.2. Constrained Alignment

We sometimes encounter a situation like that in **Fig. 3.6c**, in which an in-group sequence is to be added. A simple profile–profile alignment should not be applied to such a case, because one may overlook a close relatedness between the new sequence and a sequence (marked with an asterisk in **Fig. 3.6c**) in the existing alignment. Instead, we recommend another option:

```
% mafft-linsi --seed aligned_sequences new_sequence> output_file
```

The alignment within ***aligned_sequences*** is fixed, and an MSA consisting of both the ***aligned_sequences*** and ***new_sequence*** is created. This can be used even if many new sequences are to be added into an existing "seed" alignment, as shown in **Fig. 3.6d**.

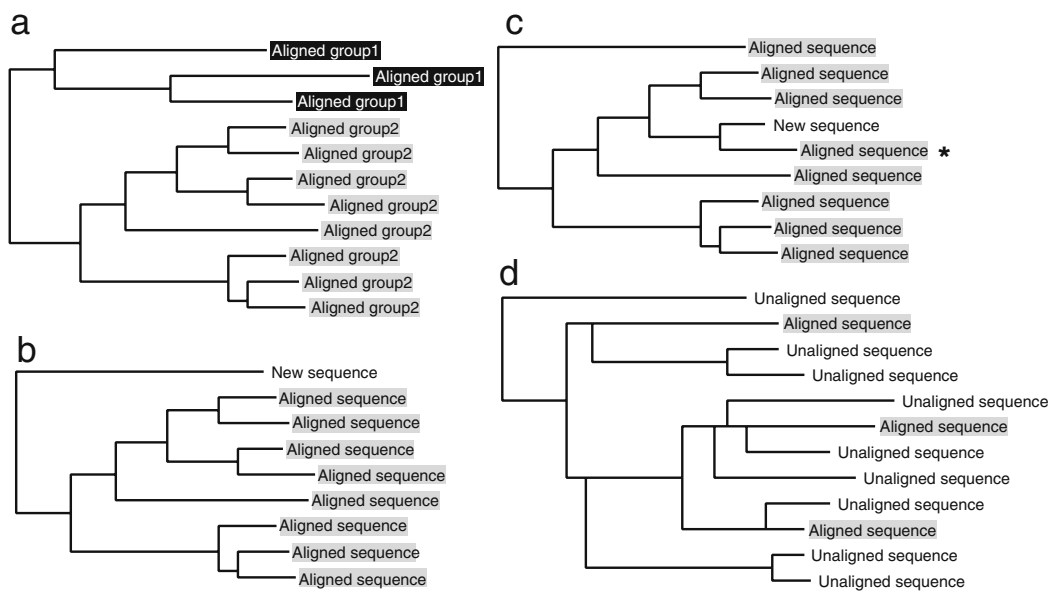


Fig. 3.6. Possible relationships between a group of aligned sequences and new sequence(s). (**a**) and (**b**) can be subjected to a profile–profile alignment, but (**c**) and (**d**) should not.

```
% mafft-linsi --seed aligned_sequences unaligned_
  sequences \ > output_file
```

Furthermore, two or more fixed groups are acceptable.

```
% mafft-linsi --seed group1 --seed group2 --seed
  group3\ unaligned_sequences> output_file
```

Users can select an appropriate algorithm, such as L-INS-i, E-INS-i, and G-INS-i, according to the characteristics of the sequences (global similarity or local similarity; *see* **Section 2.3**).

2.5. Outputting a Guide Tree

The guide tree is output by the --**treeout** argument.

```
% mafft --retree 0 --treeout --globalpair input_file>
  output_file
```

This command creates the ***input_file*.tree** file, which shows the guide tree based on pairwise alignments. The --**retree 0** option suppresses the progressive alignment calculation. Either of --**localpair**, --**globalpair** or --**genafpair** argument is recommended when the number of sequences is less than ~500.

For larger dataset (number of sequences > ~500), an approximate method based on 6mer counting is recommended, keeping in mind that the resulting tree is more approximate than that generated by the above methods.

```
% mafft --retree 0 --treeout --6merpair input_file>
  output_file
```

If the --**retree 0** option is omitted, the second guide tree (Tree 2 in **Fig. 3.2**) is output.

```
% mafft --treeout input_file> output_file
```

This tree is generally expected to be more accurate than the initial guide tree (Tree 1) as explained in **Section 1.2**. However, this is not the case when evolutionary unrelated sequences are included. This is because the second guide tree is built based on an MSA of unrelated sequences and such MSA may contain erroneous information, since the basic assumption that all of the sequences in the alignment are homologous is violated (*see* **Section 1.1**). In such a case, the use of the --**retree 0** argument is recommended. *See* **Section 3.3** for detailed discussion.

2.6. Genome Alignment

If the input sequences have translocations, inversions, or duplications, MAFFT is not suitable; MSA tools specially designed for such data should be used instead. In this section we focus on two genome alignment tools: TBA and MAUVE.

TBA *(7, 8)* is an MSA tool that can also be used to align whole genomes, though this is tractable on a single computer only for genomes that are at most a few million bases long. The latest revision (available on the web) can detect all types of genomic rearrangements. It outputs alignment blocks, where each block can contain any subset of the input species. An alignment block

cannot contain a species more than once and, by default, any genomic base of the input can appear in at most one alignment block in the output.

The first step in utilizing TBA is to align all pairs of genomes with BLASTZ *(36)*, a BLAST-like pairwise alignment tool. The TBA software package provides a wrapper program that calls BLASTZ for all pairs and post-processes the results to resolve overlapping alignments. The resulting pairwise alignments are used as input to the TBA program, along with a user-supplied binary tree of the genomes that dictates the order in which progressive alignment shall be performed. The algorithm works recursively: for each node of the tree, a set of alignment blocks corresponding to the multiple alignment of the genomes of the left subtree, and a set of alignment blocks corresponding to the multiple alignment of the genomes of the right subtree are recursively obtained. The two sets are then combined using pairwise alignments between genomes of the left subtree and genomes of the right subtree as guides (this step is called MULTIZ). The result is a set of alignment blocks corresponding to the multiple alignment of all the genomes under that node. Once the algorithm reaches the root, the final output is returned in a single file using the MAF format *(37)* (**Fig. 3.7**).

TBA can also be used to perform a multiple alignment with respect to a reference genome. The UCSC Genome Browser *(38)* utilizes MULTIZ to create and display reference-based whole genome alignments, such as a human-centric alignment of various vertebrates and a *D. melanogaster*-centric alignment of various insects. All such whole-genome MULTIZ alignments can be viewed in graphic and text form and downloaded directly from the UCSC Genome Browser website.

Both the TBA and BLASTZ packages can be downloaded from the CCGB website. To make an alignment with TBA, the input sequences should be in the FASTA format, and each species should be placed in a file of its own, named after the species (with no filename extension). The FASTA header of each sequence needs to have the following format:

>***species_name***:***chromosome_name***:***start_position***:
orientation: ***chr_size***

```
a score=4622.0
s crescentus  277993 61 + 4016947 GCAAAACGTCGCTATAAGGCGCCATGTCAAACCAAGAAAAAGCAATCGCCGTCGTCGAACG
s k31         365323 61 + 5876911 GCGGAACGCTGATATAGGGCGCCATGTCAAACCAAGAAAAAGCAATCGCCGCTGTCGAGCG

a score=7121.0
s crescentus  278054 53 + 4016947 GCTCAATGACGAATACGAACGCGCCGTCGGCGCTCTGCGCGACGCGCTCCGCG
s maris      1082062 53 - 3364850 GCTCAATGACCGCTTTGCCCGCGCCATCGACACGCTGCTGGCCCCGCTGCACG
s k31         365384 53 + 5876911 GCTAAACCAGGAATACGAACGCGCCGTCGACGCCCTGCGGACCGCGCTTCGCG

a score=2608.0
s crescentus  278107 47 + 4016947 CCTATCTCGAAGATGGAACCCGCCCGGATCCGGCCGCACGCTTTGAC
s k31         365437 47 + 5876911 CCTATCTCGAACATGGAACCCGGCCCGATCCGCAGACGCGGTTCGAC
```

Fig. 3.7. An example of alignment blocks generated by TBA from three bacterial genomes.

In this format, ***species_name*** and ***chromosome_name*** are the names of the species and the chromosome to which the sequence corresponds, and ***chr_size*** is the original size of that chromosome. In case the sequence is a piece of a chromosome (for example, if a smaller region has been extracted from a larger chromosome in order to make an alignment of a specific locus), ***start_position*** and ***orientation*** refer to the beginning position (1-indexed, inclusive) and the orientation (the ASCII character + or −) of the piece (otherwise they should be given as **1** and +, respectively). If a species contains only one sequence in its respective FASTA file, then its header can be simplified to just the species name; otherwise, if a species consists of many sequences, the special header format must be used.

To speed up BLASTZ, and also to remove spurious pairwise alignments from the BLASTZ output, low-complexity regions as well as repeats should be *softmasked* (that is, converted to lowercase characters, with the rest of the genome in uppercase) in the input sequences. This can be achieved by running RepeatMasker *(39)* and TRF *(40)* on the genomes. Repeats cover almost 50% of the human genome, so this step is essential for repeat-rich sequences. Bacterial genomes are less prone to this problem as they are not so repetitive, but it is still a good practice to mask any low complexity regions they might contain.

With the input sequences softmasked and in their proper form, the following two commands should be used to make an alignment with default parameters:

```
% all_bz "(guide tree)"
% tba "(guide tree)" *.maf output_file
```

The "(***guide tree***)"is a parenthesized expression that describes a binary tree of the species, such as "**(((human chimp) (mouse rat)) chicken)**"; it also serves as a list of filenames, since each genome is expected to reside in a file named after its name. The generated ***output_file*** will contain the resulting alignments in MAF format (**Fig. 3.7**). These can be visualized with Gmaj *(41)*, an interactive viewer (written in Java) for MAF files.

To make an alignment with respect to a reference genome, **F**=***genome*** should be added to the all bz call, and **E**=***genome*** **P**=**multic** to the tba call (where ***genome*** is the name of the genome used as reference):

```
% all_bz F=genome"(guide tree)"
% tba E=genome P=multic "(guide tree)" *.maf output_file
```

The all_bz call can also take an extra argument: the name of a file that describes the parameters of the underlying BLASTZ calls for each pair of genomes. By specifying such a file, different parameters can be used for each genomic pair, allowing, for example, more sensitive parameters for distant pairs. More documentation on this feature is available on the web *(42)*.

MAUVE *(9, 10)* is a multiple genome alignment and visualization tool available for various platforms. It does not require an external BLAST-like program, as it employs its own method for finding matching words (seeds) between the genomes, and extending them into larger anchors. Collinear anchors between genomes are combined into locally collinear blocks (LCBs). A multiple global alignment is then performed for each significant LCB, and the results are output to a file using the XMFA format *(43)* and visualized in-place with a bundled Java application. Following the traditional paradigm, MAUVE aligns each genomic base at most once in the output. Although the initially published version *(9)* detected genomic rearrangements, it also required all of the species to be present in an alignment block; a newer version called Progressive MAUVE (available on the web) allows for the generation of alignment blocks of any subset of species.

MAUVE can be easily run through its graphical front-end. Unless the sequences compared are very closely related, the progressive version of the algorithm should be used in order to achieve greater sensitivity and relax the constraint requiring the presence of *all* species in a block. The progressive alignment dialog box can be used to select the FASTA sequences to be aligned, and also to tweak the alignment parameters. The *Match Seed Weight* refers to the word size (more precisely, the number of matching positions in the seed) used to look for anchors. If the program cannot find any anchors with the default parameters due to low sequence identity, and assuming

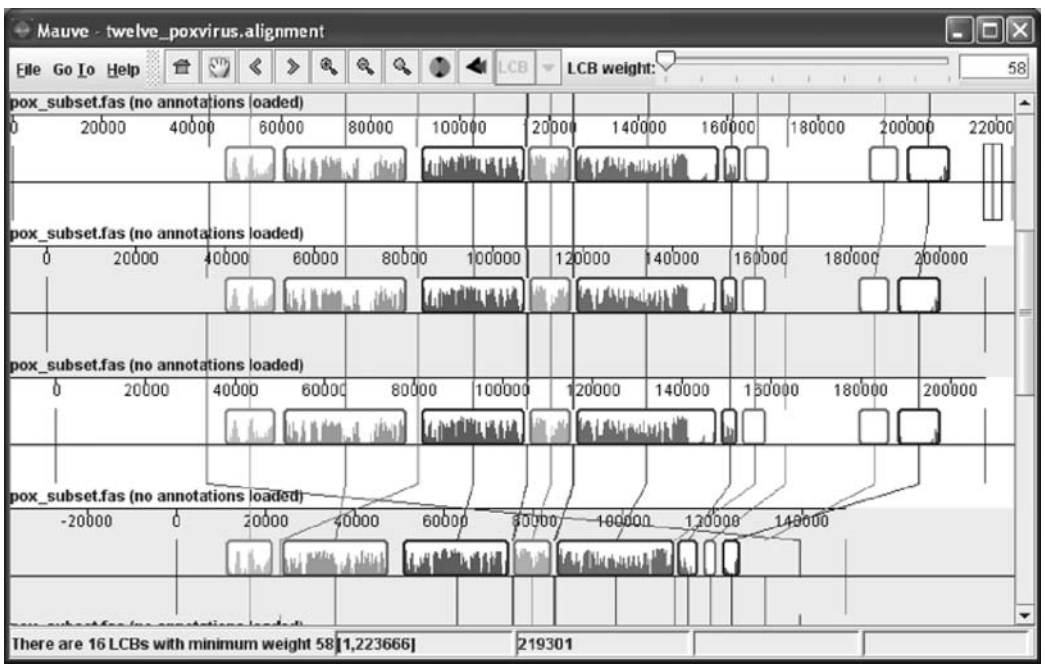


Fig. 3.8. A screenshot of MAUVE.

the genomes are small enough and not repetitive, the seed weight can be gradually lowered to permit a more sensitive anchoring. The *Minimum LCB Weight* refers to the minimum required pairwise score of an LCB. The default value (which is calculated on the fly and reported in the console output) may need to be manually increased for very closely related (nearly identical) species, or decreased for distant genomes with a lot of rearrangements and fragmented assemblies. Once the alignment is done, it will be automatically visualized (**Fig. 3.8**). The Java application can be also used to visualize previous alignments by loading the respective XMFA file.

3. Examples

3.1. Constructing an MSA of 17 Eukaryotic LSU rRNA Sequences

Here we show how to use MAFFT to construct an MSA of LSU rRNA sequences as an example. The dataset is downloadable from http://align.bmr.kyushu-u.ac.jp/mafft/examples/lsu_euk. The lsu_euk file contains 17 LSU rRNA sequences from various Eukaryotes, including animals, fungi, plants, and protists, with sequence lengths of ~3500 (*Tetrahymena*) to ~5000 (human) residues. The sequence file was subjected to the FFT-NS-2 option of MAFFT. A part of the resulting alignment is shown in **Fig. 3.9**. Most of other options, such as FFT-NS-i and L-INS-i, are applicable to this dataset, but the difference in alignment quality is small in such a highly conserved dataset. The CPU times of FFT-NS-2 and L-INS-i were ~9 s and ~2 min, respectively, on a 3 GHz Intel Xeon processor with Mac OS X. A rough phylogenetic tree of the LSU rRNA sequences by the NJ method is shown in **Fig. 3.10**.

3.2. Adding a Sequence to an Existing MSA

Assume that a partial LSU rRNA sequence (1439 residues) of chimpanzee has been newly determined. The chimpanzee sequence is at http://align.bmr.kyushu-u.ac.jp/mafft/examples/lsu_chimp. We explain here how to add the chimpanzee sequence into the existing alignment of 17 Eukaryotes. Some consideration is needed if the new sequence is expected to be closely related to one of existing sequences, like human sequence in this case (**Fig. 3.10**).

The **mafft-profile** program, which performs a group-to-sequence alignment or a group-to-group alignment, is very fast (requires less than 1 s) but not appropriate for this case. This is because **mafft-profile** converts the alignment consisting of 17 sequences into a profile ignoring the close relationship between chimpanzee and human sequences. **Figure 3.9b** shows an example of misalignment introduced by an incorrect use of **mafft-profile**. To avoid such misalignment, either of the following two procedures is recommended.

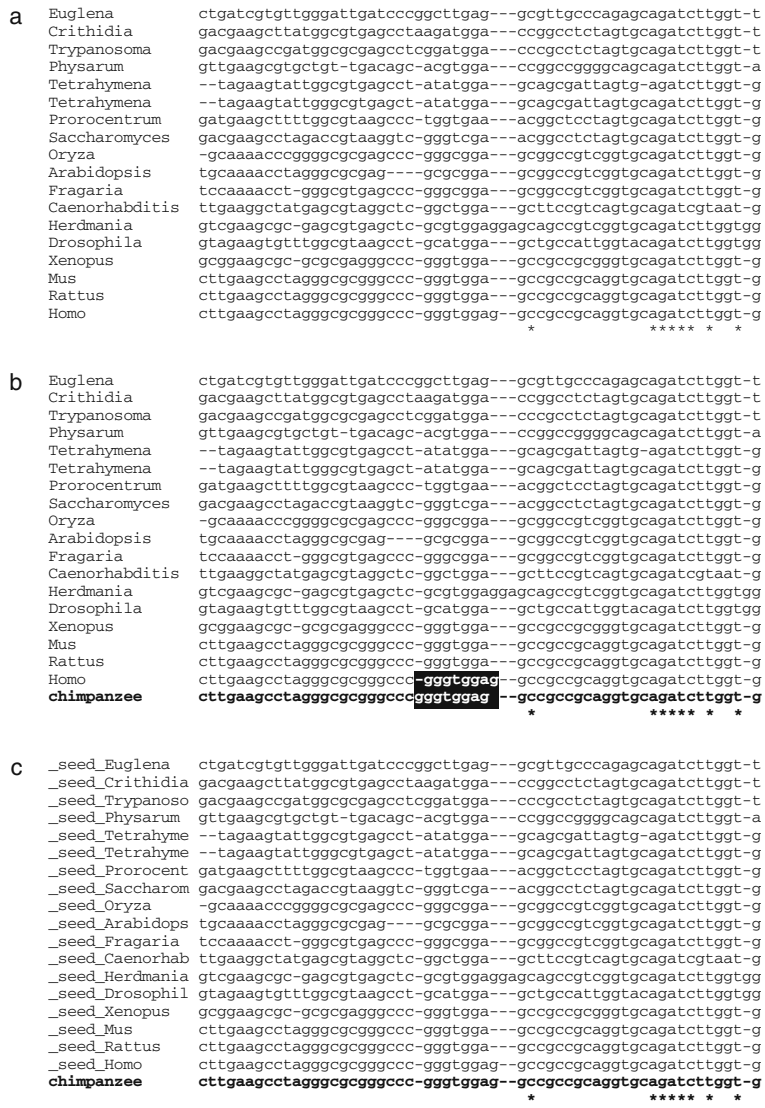


Fig. 3.9. Alignments of eukaryotic LSU rRNA sequences. (**a**), a part of the FFT-NS-2 alignment consisting of 17sequences; (**b**), an example of misalignment (highlighted) when adding the chimpanzee sequence (in bold letters) with an incorrect use of mafft-profile; (**c**), the chimpanzee sequence was added by the **--seed** option.

- If the original alignment was fully automatically constructed, remove all of the gaps from the 17 sequences and then apply the same program to the 17 (original) + 1 (chimpanzee) dataset to entirely re-construct an alignment consisting of the 18 sequences. The CPU times of FFT-NS-2 and L-INS-i are ~10 s and ~2 min, respectively, for this example.
- If the original alignment has already been manually inspected, the entire re-construction is not practical. Instead, the constrained alignment, explained in **Section 2.4**, can be useful. **Figure 3.9c** shows a resulting alignment in which a

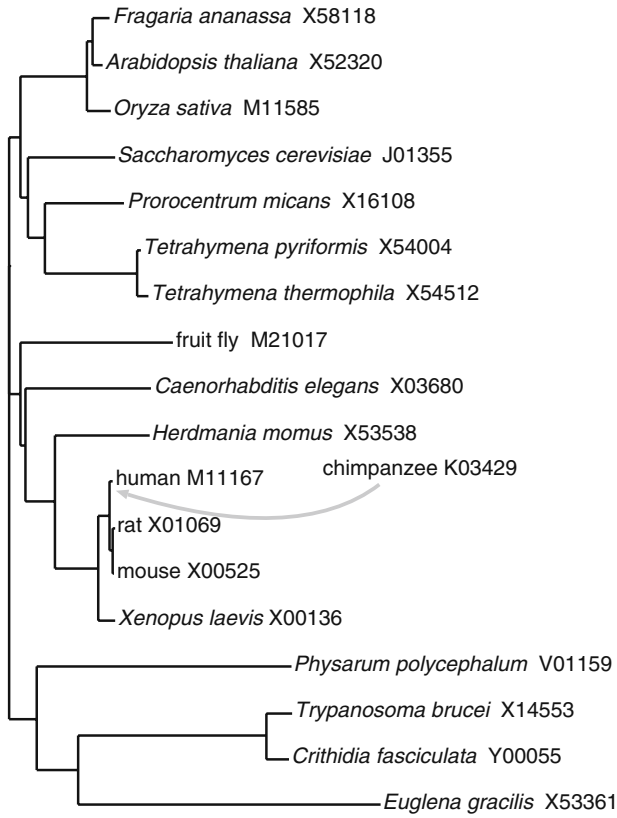


Fig. 3.10. Eukaryotic LSU rRNA sequences used in the example. Genbank accession numbers are shown after each species name. Note that this is a rough NJ tree just to show what organisms are included and is expected to reflect the phylogenetic relationship to some extent, but not completely.

chimpanzee sequence was added into the 17-species alignment by applying the constrained alignment option (**mafft --seed**). The FFT-NS-2 option was applied using the original alignment of the 17 sequences as a "seed." CPU time was ~17 s. The L-INS-i option can also be applied and it took ~1 min.

3.3. Determining Which Sequences Should be Included in an MSA

The third example is related to a problem as to which sequences should be included in an MSA and which sequences should not. We use here an artifactual example of heterogeneous data consisting of 32 sequences, including unrelated ones, SSU rRNA, tRNA, and randomly selected cDNA sequences. The dataset is available at http://align.bmr.kyushu-u.ac.jp/mafft/examples/unrelated. The SSU rRNA sequences used here (~1500 to ~2000 residues in length) were taken from all three domains, Eukarya, Bacteria, and Archaea, and they have sequence similarity. The tRNA sequences (~70 residues in length) have very weak sequence similarity to each

other and no homology to the SSU rRNA sequences. The rest were randomly taken from mouse cDNA sequences with sequence lengths of ~100 to ~700 residues. How can we graphically display the relatedness, or correctly extract the two homologous groups (SSU rRNA and tRNA)?

The most general criterion to determine whether a sequence should be included into an MSA is *E*-value calculated by BLAST *(44)* or other search tools. The sequences showing lower *E*-values to a query than a given threshold (for example, $E \leq 10^{-5}$) can be presumed as homologs and then be included into an MSA. However, such criterion is not relevant when we need an MSA of distantly related homologs, such as tRNA sequences. In this example, the *E*-values between tRNA sequences are 1 or larger and it cannot discriminate tRNA sequences from non-homologous sequences as far as standard similarity search tools are used.

In such a case, a guide tree calculated at the initial step of MSA is sometimes useful. A guide tree can show the relationship among many sequences at a glance, regardless of whether the sequences are evolutionarily related. It can be used to roughly classify (unrelated and related) sequences or to identify closely related subsets. **Figure 3.11a** shows a guide tree used in the initial step of the G-INS-i method. It was calculated with the **--treeout --globalpair --retree 0** arguments. CPU time was ~11 s. Two clusters, SSU rRNA and tRNA, are successfully discriminated. More approximate guide tree based on the 6mer counting method (**--treeout --6merpair --retree 0**) is shown in **Fig. 3.11b**. This method is very fast (requires CPU time of ~0.4 s for this example) and thus has a merit in processing larger number (~1000 to ~10,000) of sequences at the cost of accuracy.

Note that these two procedures do not include the calculation of MSA of unrelated sequences, and thus are applicable to a dataset containing evolutionarily unrelated sequences. On the other hand, the second guide tree in the progressive method (Tree 2 in **Fig. 3.2**) is not suitable for the present case with unrelated sequences because the resulting tree is built based on an MSA of unrelated sequences. Therefore, when the input sequences include evolutionary unrelated sequences, the **--tree out** option should be used always with the **--retree 0** option. **Figure 3.11c** shows the second guide tree calculated with an incorrect use of the **--treeout** option.

The guide trees explained in this section reflect the phylogenetic relationship to some extent, but it should be noted that their detailed branching order is not reliable at all. This is partly because the compared regions are not identical for pairs; almost an entire region is compared for a pair of closely related sequences, whereas only highly conserved regions are compared for a pair of distantly related sequences. Such a comparison can be useful at the early stage of a comparative study for roughly classifying sequences, but the guide tree should not be interpreted as an evolutionary tree. Only the

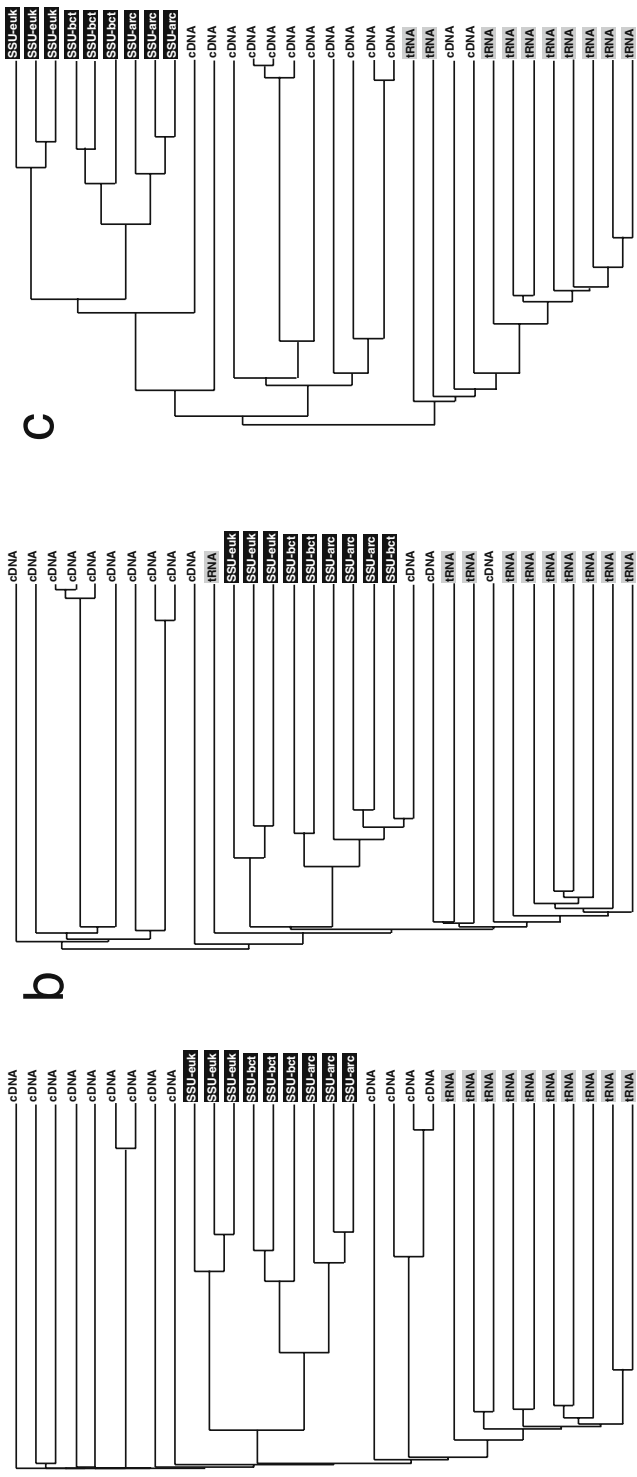


Fig. 3.11. Three types of guide trees calculated from an artifactual dataset containing unrelated sequences. SSU rRNAs are highlighted in black, tRNAs are shadowed and the rest are randomly selected cDNA sequences. (**a**), a tree based on pairwise Smith-Waterman alignments (**--globalpair**; recommended); (**b**), a tree based on 6mer distances (**--6 merpair --retree 0**; recommended only for a large dataset); (**c**), the second guide tree (**--6 merpair**; not recommended).

members of a distinct cluster, like each of the SSU and tRNA clusters in **Fig. 3.11a** and **b**, can be subjected to the MSA methods explained in **Sections 2.1, 2.2, and 2.3**. Once an MSA of homologous sequences is obtained, the MSA has much more relevant phylogenetic information than the guide tree or pairwise alignments, and the MSA can be subjected to phylogenetic inferences and/or other analyses.

4. Notes

1. Looking at **Fig. 3.2**, users who want to use an MSA for phylogeny inference may think this procedure is somewhat problematic, because guide trees, which can be thought of as an approximate distance-based phylogenetic tree, are used before an MSA is created. Will the final phylogenetic tree be affected by the initial guide tree? In our opinion, this problem makes little bias on the phylogenetic inference, as long as only unambiguously aligned sites are subjected to the phylogenetic analysis. One way to determine unambiguously aligned sites is as follows: (1) create two or more MSAs by independently applying different tools to the same set of unaligned sequences, (2) compare the MSAs using a tool such as ALTAVIST *(45)* and Mumsa *(46)*, and then (3) extract the regions of agreement. The resulting sites are more likely to be correctly aligned than those generated by an MSA with a single tool. This procedure improves the specificity of the alignment, but of course, at the cost of sensitivity. Iterative refinement methods also use a guide tree as shown in **Fig. 3.3**, and thus have the same problem discussed earlier: the alignment and the final phylogenetic tree might be affected by the guide tree. The doubly nested iterative refinement method of PRRN *(25)* is another possible solution to this problem.
2. Some MSA methods assume that all of the input sequences are globally alignable; that is, the overall sequences (5′ to 3′) are assumed to be homologous, but this assumption does not necessarily agree with real analyses. To overcome this problem, some other MSA methods incorporate a local alignment algorithm *(4, 30, 47)* to allow large successive gaps at both termini. Local alignment methods avoid the assumption of global homology by identifying only short patches of strong sequence similarity. Some methods maintain the assumption of global homology but impose different penalties for terminal gaps in order to allow for some degree of local alignment character *(16, 48–50)*.
3. TCoffee *(30)* and ProbCons *(49)* use consistency information as follows. This process is called "library extension" in TCoffee. Given three sequences, A, B, and C, all of the pairwise alignments

A-B, B-C, and A-C are calculated. The collection of these alignments is called a "primary library." An alignment between A-B can also be calculated by synthesizing A-C and B-C, and it may differ from an alignment generated by a direct comparison between A and B. This process is carried out for all possible combinations of the three sequences from the input data. The resulting collection of pairwise alignments is called an "extended library." The next step is progressive alignment, in which a group-to-group alignment is performed using a DP matrix whose elements are calculated as the summation of the scores of pairwise alignments in the library that has the matching corresponding to the element. As a result, in a group-to-group alignment step, two (groups of) sequences, say A and B, are aligned while taking into account the information of other sequences, say C.

4. Most alignment methods have several parameters, including a scoring matrix that describes the relative frequency of each substitution type and gap penalties that are related to the frequency of insertion/deletion events. The parameters, especially the gap penalties, greatly affect the alignment quality. The default parameters of most MSA methods are tuned by trial-and-error, but some methods systematically determine the parameters from unaligned sequences in an unsupervised manner *(49)*. As for MAFFT, we recommend, if possible, trying out several sets of entirely different gap penalties, such as

   ```
   % mafft-linsi --op 1.53 --ep 0.123 input_file> output_file
   ```

   (equivalent to default of mafft-linsi)

   ```
   % mafft-linsi --op 2.4 --ep 0.0 --fmodel input_file> output_file
   % mafft-ginsi --op 2.4 --ep 0.5 input_file> output_file
   ```

5. A large number of MSA programs have been developed as listed in **Table 3.2**. It is highly recommended to use multiple different methods to obtain a high quality alignment *(11*, *51*, *52)*. A decision tree for selecting appropriate tools (**Fig. 3.5** in ref. *53)* in various situations may be useful. The alignments generated by different methods can be combined into a single MSA by meta-aligners, such as MCoffee *(51)* in the TCoffee package. A combined MSA is generally more reliable than each MSA calculated by a stand-alone program *(51)*. In addition, Mumsa *(46)* and similar tools use multiple independent MSAs to improve the specificity (*see* **Note 1**).

6. There are also many visualization tools (**Table 3.3**) that allow interactive alignment editing and manual identification of reliably aligned regions. For alignments of sequences with

low-similarity, manual inspection is extremely important. Morrison *(54)* addressed various situations in which the mathematically optimal alignment is not a biologically correct alignment.

Table 3.2
MSA programs

ClustalW *(16)*	ftp://ftp.ebi.ac.uk/pub/software/clustalw2/
PRRN *(25)*	http://www.genome.ist.i.kyoto-u.ac.jp/~aln_user
DIALIGN *(45)*	http://bibiserv.techfak.uni- bielefeld.de/dialign/
TCoffee *(30)*	http://www.tcoffee.org/
MAFFT *(3)*	http://align.bmr.kyushu- u.ac.jp/mafft/software/
MUSCLE *(48)*	http://www.drive5.com/muscle/
ProbConsRNA *(49)*	http://probcons.stanford.edu/
Kalign *(50)*	http://msa.cgb.ki.se/
ProbAlign *(55)*	http://www.cs.njit.edu/usman/probalign/
PRIME *(56)*	http://prime.cbrc.jp/
MLAGAN *(57)*	http://lagan.stanford.edu/lagan_web/index.shtml
MAVID *(58)*	http://baboon.math.berkeley.edu/mavid/
MUMmer [59]	http://mummer.sourceforge.net/
CCGB (TBA and BLASTZ)	http://www.bx.psu.edu/miller_lab
MAUVE *(9)*	http://gel.ahabs.wisc.edu/mauve/

Table 3.3
Alignment viewers

Jalview	http://www.jalview.org/
Kalignvu	http://msa.cgb.ki.se/
ClustalX	ftp://ftp.ebi.ac.uk/pub/software/clustalw2
UCSC	http://genome.ucsc.edu/
Gmaj	http://globin.cse.psu.edu/dist/gmaj/

Acknowledgments

We thank Chuong B. Do for critical reading of the manuscript.

References

1. Woese, C. R., and Fox, G. E. (1977) Phylogenetic structure of the prokaryotic domain: the primary kingdoms. *Proc Natl Acad Sci USA* **74**, 5088–90.
2. Flicek, P., Keibler, E., Hu, P., Korf, I., and Brent, M. R. (2003) Leveraging the mouse genome for gene prediction in human: from whole-genome shotgun reads to a global synteny map. *Genome Res* **13**, 46–54.
3. Katoh, K., Misawa, K., Kuma, K., and Miyata, T. (2002) MAFFT: a novel method for rapid multiple sequence alignment based on fast Fourier transform. *Nucleic Acids Res* **30**, 3059–66.
4. Katoh, K., Kuma, K., Toh, H., and Miyata, T. (2005) MAFFT version 5: improvement in accuracy of multiple sequence alignment. *Nucleic Acids Res* **33**, 511–8.
5. Wilm, A., Mainz, I., and Steger, G. (2006) An enhanced RNA alignment benchmark for sequence alignment programs. *Algorithms Mol Biol* **1**, 19.
6. Carroll, H., Beckstead, W., O'connor, T., Ebbert, M., Clement, M., Snell, Q., and McClellan, D. (2007) DNA reference alignment benchmarks based on tertiary structure of encoded proteins. *Bioinformatics* **23**, 2648–49.
7. Blanchette, M., Kent, W. J., Riemer, C., Elnitski, L., Smit, A. F., Roskin, K. M., Baertsch, R., Rosenbloom, K., Clawson, H., Green, E. D., Haussler, D., and Miller, W. (2004) Aligning multiple genomic sequences with the threaded blockset aligner. *Genome Res* **14**, 708–15.
8. http://www.bx.psu.edu/miller_lab
9. Darling, A. C., Mau, B., Blattner, F. R., and Perna, N. T. (2004) Mauve: multiple alignment of conserved genomic sequence with rearrangements. *Genome Res* **14**, 1394–403.
10. http://gel.ahabs.wisc.edu/mauve/
11. Edgar, R. C., and Batzoglou, S. (2006) Multiple sequence alignment. *Curr Opin Struct Biol* **16**, 368–73.
12. Needleman, S. B., and Wunsch, C. D. (1970) A general method applicable to the search for similarities in the amino acid sequence of two proteins. *J Mol Biol* **48**, 443–53.
13. Smith, T. F., and Waterman, M. S. (1981) Identification of common molecular subsequences. *J Mol Biol* **147**, 195–7.
14. Gotoh, O. (1982) An improved algorithm for matching biological sequences. *J Mol Biol* **162**, 705–8.
15. Feng, D. F., and Doolittle, R. F. (1987) Progressive sequence alignment as a prerequisite to correct phylogenetic trees. *J Mol Evol* **25**, 351–60.
16. Thompson, J. D., Higgins, D. G., and Gibson, T. J. (1994) CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. *Nucleic Acids Res* **22**, 4673–80.
17. Katoh, K., and Toh, H. (2007) Parttree: an algorithm to build an approximate tree from a large number of unaligned sequences. *Bioinformatics* **23**, 372–4.
18. Barton, G. J., and Sternberg, M. J. (1987) A strategy for the rapid multiple alignment of protein sequences. confidence levels from tertiary structure comparisons. *J Mol Biol* **198**, 327–37.
19. Berger, M. P., and Munson, P. J. (1991) A novel randomized iterative strategy for aligning multiple protein sequences. *Comput Appl Biosci* **7**, 479–84.
20. Gotoh, O. (1993) Optimal alignment between groups of sequences and its application to multiple sequence alignment. *Comput Appl Biosci* **9**, 361–70.
21. Ishikawa, M., Toya, T., Hoshida, M., Nitta, K., Ogiwara, A., and Kanehisa, M. (1993) Multiple sequence alignment by parallel simulated annealing. *Comput Appl Biosci* **9**, 267–73.
22. Notredame, C., and Higgins, D. G. (1996) Saga: sequence alignment by genetic algorithm. *Nucleic Acids Res* **24**, 1515–24.
23. Gotoh, O. (1994) Further improvement in methods of group-to-group sequence alignment with generalized profile operations. *Comput Appl Biosci* **10**, 379–87.
24. Gotoh, O. (1995) A weighting system and algorithm for aligning many

phylogenetically related sequences. *Comput Appl Biosci* **11**, 543–51.

25. Gotoh, O. (1996) Significant improvement in accuracy of multiple protein sequence alignments by iterative refinement as assessed by reference to structural alignments. *J Mol Biol* **264**, 823–38.
26. Hirosawa, M., Totoki, Y., Hoshida, M., and Ishikawa, M. (1995) Comprehensive study on iterative algorithms of multiple sequence alignment. *Comput Appl Biosci* **11**, 13–18.
27. Vingron, M., and Argos, P. (1989) A fast and sensitive multiple sequence alignment algorithm. *Comput Appl Biosci* **5**, 115–21.
28. Gotoh, O. (1990) Consistency of optimal sequence alignments. *Bull Math Biol* **52**, 509–25.
29. Notredame, C., Holm, L., and Higgins, D. G. (1998) COFFEE: an objective function for multiple sequence alignments. *Bioinformatics* **14**, 407–22.
30. Notredame, C., Higgins, D. G., and Heringa, J. (2000) T-coffee: a novel method for fast and accurate multiple sequence alignment. *J Mol Biol* **302**, 205–17.
31. Higgins, D. G., and Sharp, P. M. (1988) CLUSTAL: a package for performing multiple sequence alignment on a microcomputer. *Gene* **73**, 237–44.
32. Jones, D. T., Taylor, W. R., and Thornton, J. M. (1992) The rapid generation of mutation data matrices from protein sequences. *Comput Appl Biosci* **8**, 275–82.
33. Altschul, S. F. (1998) Generalized affine gap costs for protein sequence alignment. *Proteins* **32**, 88–96.
34. Myers, E. W., and Miller, W. (1988) Optimal alignments in linear space. *Comput Appl Biosci* **4**, 11–17.
35. Gribskov, M., McLachlan, A. D., and Eisenberg, D. (1987) Profile analysis: detection of distantly related proteins. *Proc Natl Acad Sci USA*, **84**, 4355–58.
36. Schwartz, S., Kent, W. J., Smit, A., Zhang, Z., Baertsch, R., Hardison, R. C., Haussler, D., and Miller, W. (2003) Human-mouse alignments with BLASTZ. *Genome Res* **13**, 103–7.
37. http://genome.ucsc.edu/FAQ/FAQformat
38. http://genome.ucsc.edu/
39. Smit, A. F. A., Hubley, R., and Green, P. Repeatmasker. http://www.repeatmasker.org/
40. Benson, G. (1999) Tandem repeats finder: a program to analyze DNA sequences. *Nucleic Acids Res* **27**, 573–80.
41. http://globin.cse.psu.edu/dist/gmaj/
42. http://www.bx.psu.edu/miller_lab/dist/tba_howto.pdf
43. http://gel.ahabs.wisc.edu/mauve/mauve-user- guide/
44. Altschul, S. F., Madden, T. L., Schaffer, A. A., Zhang, J., Zhang, Z., Miller, W., and Lipman, D. J. (1997) Gapped BLAST and PSI-BLAST: a new generation of protein database search pro grams. *Nucleic Acids Res* **25**, 3389–402.
45. Morgenstern, B., Goel, S., Sczyrba, A., and Dress, A. (2003) Altavist: comparing alternative multiple sequence alignments. *Bioinformatics* **19**, 425–6.
46. Lassmann, T., and Sonnhammer, E. L. (2007) Automatic extraction of reliable regions from multiple sequence alignments. *BMC Bioinformat* **8** Suppl 5, S9.
47. Morgenstern, B., Dress, A., and Werner, T. (1996) Multiple DNA and protein sequence alignment based on segment-to-segment comparison. *Proc Natl Acad Sci USA* **93**, 12098–103.
48. Edgar, R. C. (2004) MUSCLE: multiple sequence alignment with high accuracy and high throughput. *Nucleic Acids Res* **32**, 1792–7.
49. Do, C. B., Mahabhashyam, M. S., Brudno, M., and Batzoglou, S. (2005) ProbCons: probabilistic consistency-based multiple sequence alignment. *Genome Res* **15**, 330–40.
50. Lassmann, T., and Sonnhammer, E. L. (2005) Kalign – an accurate and fast multiple sequence alignment algorithm. *BMC Bioinformat* **6**, 298.
51. Wallace, I. M., O'Sullivan, O., Higgins, D. G., and Notredame, C. (2006) M-Coffee: combining multiple sequence alignment methods with t-coffee. *Nucleic Acids Res* **34**, 1692–9.
52. Golubchik, T., Wise, M. J., Easteal, S., and Jermiin, L. S. (2007) Mind the gaps: Evidence of bias in estimates of multiple sequence alignments. *Mol Biol Evol* **24**, 2433–42.
53. Do, C. B., and Katoh, K. (2008) Protein multiple sequence alignment Functional Proteomics, *Methods Mol Biol* **484**, 379–413.
54. Morrison, D. (2006) Multiple sequence alignment for phylogenetic purposes. *Aust Syst Bot* **19**, 479–539.

55. Roshan, U., and Livesay, D. R. (2006) Probalign: multiple sequence alignment using partition function posterior probabilities. *Bioinformatics* **22**, 2715–21.
56. Yamada, S., Gotoh, O., and Yamana, H. (2006) Improvement in accuracy of multiple sequence alignment using novel group-to-group sequence alignment algorithm with piecewise linear gap cost. *BMC Bioinformat* **7**, 524.
57. Brudno, M., Do, C. B., Cooper, G. M., Kim, M. F., Davydov, E., Green, E. D., Sidow, A., and Batzoglou, S. (2003) LAGAN and multi-LAGAN: efficient tools for large-scale multiple alignment of genomic DNA. *Genome Res* **13**, 721–31.
58. Bray, N., and Pachter, L. (2004) MAVID: constrained ancestral alignment of multiple sequences. *Genome Res.* **14**, 693–9.

Chapter 4

SeqVis: A Tool for Detecting Compositional Heterogeneity Among Aligned Nucleotide Sequences

Lars Sommer Jermiin, Joshua Wing Kei Ho, Kwok Wai Lau, and Vivek Jayaswal

Abstract

Compositional heterogeneity is a poorly appreciated attribute of aligned nucleotide and amino acid sequences. It is a common property of molecular phylogenetic data, and it has been found to occur across sequences and/or across sites. Most molecular phylogenetic methods assume that the sequences have evolved under globally stationary, reversible, and homogeneous conditions, implying that the sequences should be compositionally homogeneous. The presence of the above-mentioned compositional heterogeneity implies that the sequences must have evolved under more general conditions than is commonly assumed. Consequently, there is a need for reliable methods to detect under what conditions alignments of nucleotides or amino acids may have evolved. In this chapter, we describe one such program. ***SeqVis*** is designed to survey aligned nucleotide sequences. We discuss *pros-et-cons* of this program in the context of other methods to detect compositional heterogeneity and violated phylogenetic assumptions. The benefits provided by ***SeqVis*** are demonstrated in two studies of alignments of nucleotides, one of which contained 7542 nucleotides from 53 species.

Key words: Compositional heterogeneity, nucleotide sequences, visualization, matched-pairs tests, cluster analysis, SeqVis.

1. Introduction

1.1. Compositional Heterogeneity

Compositional heterogeneity is a property used to describe differences in the composition of quantifiable traits within and/or among genomes. Commonly, the features of interest include the composition of nucleotides in DNA or amino acids in proteins (e.g., *1–3*), but recent technological advances in the sequencing and annotation of whole genomes have led to the discovery of many other genomic traits that can be studied qualitatively and

David Posada (ed.), *Bioinformatics for DNA Sequence Analysis,* Methods in Molecular Biology 537

DOI 10.1007/978-1-59745-251-9_4

quantitatively (e.g., transposons, long terminal repeats (LTRs), short interspersed nuclear elements (SINEs), and long interspersed nuclear elements (LINEs)) *(4)*.

Even before the genome era had begun, there was evidence of compositional heterogeneity within and among bacterial genomes *(5–9)* and among animal mitochondrial genomes *(1, 2, 10, 11)*. Using density-gradient centrifugation to separate long segments of DNA (according to their density), Bernardi et al. *(12)* discovered that nuclear DNA of warm-blooded vertebrates is a mosaic of isochores (i.e., long segments of DNA, each with a homogeneous composition of nucleotides and, thus, density), whereas Mouchiroud et al. *(13)* discovered that gene density is higher in the GC-rich isochores than in the GC-poor isochores (for a comprehensive review of the architecture of genomes, *see* ref. *14*). Some of these observations have subsequently been confirmed through studies of completely sequenced chromosomes (e.g., *15, 16*).

1.2. Phylogenetic Assumptions

Based on the evidence from a growing body of research (e.g., *2, 3, 17–40*), it appears that compositional heterogeneity also is present among homologous sequences of nucleotides or amino acids—in some cases, it occurs across sequences; in other cases, it occurs across sites. Either way, however, the presence of compositional heterogeneity has implications on how alignments of sequence data should be analyzed phylogenetically because most phylogenetic methods assume that individual sites in DNA or protein have evolved independently under the same stationary, reversible, and homogeneous conditions (for details, *see* refs. *35, 40–42*). In the context of a nucleotide sequence evolving on a phylogeny, the evolutionary process is as follows: *stationary* when the marginal distribution of the process is equal to the nucleotide content of the ancestral sequence; *reversible* when the probability of sampling nucleotide *i* from the stationary distribution and going to nucleotide *j* is equal to that of sampling nucleotide *j* from the stationary distribution and going to nucleotide *i*; and *homogeneous* when the conditional rates of change are constant overtime.

An evolutionary process may pertain to a single edge in the phylogeny, in which case it is a local process (e.g., a locally homogeneous process), or it may pertain to all the edges of the phylogeny, in which case it is a global process (e.g., a globally homogeneous process). The most commonly used phylogenetic methods assume that the sequences have evolved under globally stationary, reversible, and homogeneous conditions.

Interestingly, the relationship among the above-mentioned three conditions is more complex than indicated thus far because a reversible process is, by definition, also a stationary process *(43)*, implying that the evolution of diverging sequences can be described using six possible scenarios (**Table 4.1**). However,

Table 4.1
The spectrum of conditions under which diverging sequences can evolve

Scenario	Stationarity	Reversibility	Homogeneity	Comment on each scenario
1	✓	✓	✓	Possible; this is the scenario assumed by most phylogenetic methods
2	✗	✓	✓	By definition, impossible *(see refs. 40, 43)*
3	✓	✗	✓	Possible
4	✗	✗	✓	Possible
5	✓	✓	✗	Possible
6	✗	✓	✗	By definition, impossible *(see refs. 40, 43)*
7	✓	✗	✗	Possible
8	✗	✗	✗	Possible

visual inspections of alignments of homologous sequences often do not allow us to determine under what conditions the sequences may have evolved (e.g., *see* **Fig. 4.1**). Hence, given that compositional heterogeneity across sequences and/or sites

- appears to be a common characteristic of molecular phylogenetic data (e.g., *2, 3, 17–40*),
- indicates that these sequences cannot have evolved under the same globally stationary, reversible, and homogeneous conditions *(37, 40)*,
- appears to bias phylogenetic results obtained using phylogenetic methods that assume the sequences have evolved under globally stationary, reversible, and conditions (e.g., *3, 18–23, 25, 28–33, 35, 38, 40, 44–47)*,

and that:

- most phylogenetic methods assume that the sequences have evolved under globally stationary, reversible, and homogeneous conditions (i.e., they rely on the family of time-reversible Markov models that extends from a simple one-parameter model *(48)* to the general time-reversible model *(49)*),

there is a need for statistically sound methods that can distinguish between cases where sites have evolved under: (*i*) the same globally stationary, reversible, and homogeneous conditions, in which case approximation of the evolutionary processes might be done using a single time-reversible Markov model; (*ii*) different globally stationary, reversible, and homogeneous conditions, in which case approximation of the evolutionary process would require more than one time-reversible Markov model; or (*iii*) more general

```
Ajam    ATAAACATAGCCATTACACTATTAACAAATACATTTCTAGCAAGCCTCCTCGTAATAATTGCATTTTGATTACCCCAAACA
Ctub    ATTAATTTTATACTAACACTGCTCATTAATACCCTGCTGGCACTATTACTAGTAACCATTGCATTCTGACTACCCCATATG
Mtub    ATAAATATAGCATTAACATTACTTGTCAACACCCTACTGGCATCTCTATTAGTCATAATCGCATTCTGACTACCTCAGACA
Pabr    ATTAATTTTATATTAACATTATTTACCAACACCCTACTAGCACTACTGCTAGTAACAATCGCATTCTGATTACCCCATATA
Rmon    ATAAACTTTATATTAACCCTTCTAACAAACACCTTGCTAGCTCTACTGCTCGTAACTATCGCATTCTGACTCCCACAAACC
Rpum    ATAAACTTTATATTAACCCTTCTAACAAACACTTTGCTAGCTCTACTGCTCGTAACTATCGCATTCTGACTCCCACAAACC
        ** **  *     * **  *  *    ** **  * ** **     *  * ** *  ** ***** *** * ** ** *

AATTCCTATACAGAAAAAGTAAGTCCCTATGAATGTGGATTTGATCCCTTAGGATCCGCTCGCCTCCCCTTTTCAATAAAATTCTTCCT
AATATTTACACTGAAAAATCAAGTCCATACGAATGTGGTTTTGACCCCATAGGATCTGCCCGTCTTCCCTTTTCAATAAAATTTTTCTT
AACGTCTATACAGAAAAGTCAAGCCCTTACGAATGTGGATTTGACCCGATAGGCTCAGCACGTCTCCCCTTCTCTATGAAATTCTTTCT
AATACCTATGCCGAAAAATCAAGCCCGTATGAATGTGGCTTTGATCCAATGGGATCAGCTCGCCTACCATTCTCAATAAAATTCTTCTT
AACGTATACTCAGAAAAATCAAGTCCCTATGAGTGTGGATTCGATCCTATAGGATCTGCCCGTTTACCCTTTTCCATAAAATTTTTCCT
AACGTATACTCAGAAAAATCAAGCCCTTATGAATGTGGATTTGATCCTATAGGATCTGCCCGTTTACCCTTTTCCATAAAATTTTTCCT
**    **  * ***** * *** ** ** ** ***** ** ** **  * ** ** ** **  * ** ** ** ** ***** **  *

AGTAGCCATTACATTTCTTTTATTTGACCTAGAAATTGCACTTCTCCTCCCACTTCCCTGAGCATCACAAACTAACAATTTACAAAATA
AATCGCCATTACCTTCTTACTATTTGACCTGGAAATCGCACTACTATTACCCCTACCATGAGCCTCCCAAACAAACAAACTCTCAATTA
AGTAGCCATTACATTTTTATTATTTGATTTAGAAATCGCCCTACTCCTACCCCTTCCATGAGCCTCTCAAACTAATGAGCTAGCAACCA
GGTAGCCATTACATTTTTACTATTTGACCTCGAAATTGCTCTTCTACTGCCCTTACCATGAGCCTCCCAAACAGATAAACTTTTAGTTA
AGTAGCCATTACATTCCTGCTGTTTGACCTGGAAATCGCCCTACTCTTACCACTACCATGAGCATCCCAAGCCAACAACCTAGAAGTCA
AGTAGCCATTACATTCCTGCTATTTGACCTGGAAATTGCCTTACTCTTACCACTACCATGAGCATCCCAAGCCAACAACCTAGAAGTCA
  * ******** **  *  * *****  * ***** **  * **  * **  * ** ***** ** ***    *  *  *   *  *

TACTCCCAATATCTCTAATACTAATCTCTCTTTTAGCCCTCAGTCTAGCTTATGAGTGATTACAAGAAGGCCTAGAATGGTCTGAATAA
TGCTATCAATATCTCTATTTTTAATCCTACTACTAATCATTAGCCTAGCCTATGAGTGGATACAAAAAGGACTAGAATGGTCTGAATA
TACTACCAACATCACTATTCCTTATCTCCCTCCTAGCAATCAGTCTAGCATATGAATGATCTCAAAAGGGCCTAGAATGAACTGAATAA
TACTATCTATATCTCTAGCTCTAATTATATTATTAATCATTAGCCTTGCTTATGAATGAATACAAAAGGGGTTAGAATGATCTGAAT
TGCTCACCACAGCCCTGCTACTAATCTCCTTACTAGCTATCAGCTTAGCCTACGAATGATCTCAAAAAGGATTAGAGTGAACCGAATAA
TGCTCACCACAGCCCTGCTCCTAATCTCCTTACTAGCTATCAGCTTAGCCTACGAATGATCCCAAAAAGGATTAGAGTGAACCGAATAA
* **  * * * * **     * **     *  **    * **  * ** ** ** **    *** * **  **** **  * ****
```

Fig. 4.1. An alignment of NADH dehydrogenase subunit 3 genes from six bats (*Artibeus jamaicensis*, Ajam, NP_008491; *Chalinolobus tuberculatus*, Ctub, NC_002626; *Mystacina tuberculata*, Mtub, YP_220700; *Pipistrellus abramus*, Pabr, NP_976133; *Rhinolophus monoceros*, Rmon, NP_976094; *R. pumilus*, Rpum, NP_976107). Asterisks indicate constant sites.

conditions, in which case it would be necessary to use phylogenetic methods that incorporate more general models of molecular evolution *(35, 38, 44, 50–64)*.

1.3. Methods of Detecting Violated Phylogenetic Assumptions

A number of methods have been used to detect compositional heterogeneity in alignments of nucleotides or amino acids. The methods can be partitioned into six categories, four of which have been described previously (*see* ref. *46)*:

- Methods belonging to this category use Monte Carlo simulations or the multinomial distribution to generate sequence-specific distributions of the expected nucleotide or amino acid content, which are compared using tables and graphs *(49, 65)*. Due to their simplicity, the methods have some appeal, but they are of limited value for surveys of species-rich alignments (i.e., alignments with many sequences). Furthermore, they may yield biased results if invariable sites are not excluded before the survey *(46)*. Perhaps, it is for these reasons that the methods have been used infrequently (for two exceptions, *see* refs. *66, 67*).

- Methods belonging to this category test the homogeneity of an $n \times m$ contingency table, where n and m, respectively, are the number of sequences and the number of letters in the alphabets of nucleotides or amino acids *(68)*. In some cases, pairs of sequences are compared using multiple tests *(69, 70)*, while in other cases all sequences are compared in a single test *(68)*. PAUP* *(71),* a popular phylogenetic program, implements both of these strategies. Another popular phylogenetic program, Tree-Puzzle *(72, 73)*, implements a similar test where the composition of each sequence is compared to the estimated marginal distribution for the data (inferred assuming time-reversible conditions). Methods belonging to this category produce misleading results because the homology of sites in alignments is not properly accounted for and because sequence divergence and the presence of invariable sites may bias the results.
- Methods belonging to this category test the symmetry, marginal symmetry, and internal symmetry of a divergence array, where each element of the array is the frequency of a matching pair of nucleotides or amino acids in the alignment *(40)*. Developed in 1948 *(74)* and 1955 *(75)*, the matched-pairs tests of symmetry and marginal symmetry were later identified as being pertinent to studies of molecular phylogeny *(76)*. The methods did not, however, gain the recognition that they deserve (for known exceptions, *see* refs. *77–79*) before being described again *(36)*, this time together with a matched-pairs test of internal symmetry. Although these methods have not yet been extensively used, possibly because they are not widely known or because they are poorly understood (for recent exceptions, *see* refs. *34, 37, 39, 80–82*), they are the most appropriate methods for detecting violated phylogenetic assumptions because the alignment is examined using a site-by-site-based approach, implying that the homology of sites is accounted for, and because under the null hypothesis (i.e., that the sequences have evolved under stationary, reversible, and homogeneous conditions), the distributions of p-values from the three tests are unaffected by sequence divergence and the presence of invariable sites *(36, 40)*. Moreover, the methods can be used to survey species-rich alignments without loss of generality. Generalized versions of the matched-pairs test of marginal symmetry have also been developed *(36, 83)*, but although these tests are statistically sound, they do not allow us to identify offending sequence(s). Furthermore, marginal symmetry without symmetry is possible and might present a problem in phylogenetic studies *(40, 46, 76)*.
- This category comprises a single method that calculates a disparity index, I_d, based on frequencies of the i-th nucleotide or amino acid in two sequences (i.e., x_i and y_i) and on the number of sites where the two sequences differ, N_d. A Monte

Carlo simulation is then used to test whether $I_d = 0.0$, in which case the sequences are assumed to have evolved under the same conditions *(84, 85)*. The method has been shown to be better than those from the second category *(84, 85)*, but it should be used with caution as it does not consider the homology of sites appropriately while generating x_i, y_i, and N_d.

- Methods belonging to this category use relative nucleotide or amino acid frequencies to calculate a compositional distance (δ) between pairs of sequences in the alignment *(44)* or a relative compositional variability (*RCV*) index for all sequences in the alignment *(86)*. Citing the Euclidean distance, Lockhart et al. *(44)* defined δ_{ij} as the square root of the sum of squares of compositional differences between sequence *i* and sequence *j* in the alignment. On the other hand, Phillips and Penny *(86)* defined *RCV* as the sum of absolute sequence- and character-specific deviations from the average composition of all sequences, divided by the number of sequences and number of sites (for a similar index of compositional variation, *see* ref. *87*). Although the simplicity of these methods renders them attractive, their usefulness may be limited because null distributions for δ_{ij} and *RCV* are unknown, implying that it is impossible to determine whether an observation is large enough to be of concern. Methods belonging to this category furthermore produce misleading results because the homology of sites in alignments is not appropriately accounted for and because the results may become biased by sequence divergence.
- This category comprises a single method of displaying nucleotide content in one, two, or three dimensions *(37)*. The tetrahedral plot has been used to (*i*) study compositional patterns in codon usage *(88)* and rRNA secondary structure categories *(89)*, (*ii*) display compositional heterogeneity among homologous nucleotide sequences *(27)*, and (*iii*) detect violation of phylogenetic assumptions *(37, 40)*. The method is particularly useful for surveying species-rich alignments, but it suffers from the same shortcomings that methods belonging to the fifth category suffer from. Consequently, it is best to use this method in conjunction with the matched-pairs test of symmetry (i.e., a method belonging to the third category).

In summary, there are several methods to detect compositional heterogeneity in alignments of nucleotides and amino acids but due to the manner in which compositional heterogeneity is detected, only a combination of some of these methods meets the criteria of being able to (*i*) survey species-rich alignments of nucleotides; (*ii*) produce easy-to-interpret plots from these data; and (*iii*) provide a statistically sound corroboration of visual evidence of compositional heterogeneity (i.e., through the use of the matched-pairs test of symmetry).

In the next section, we describe ***SeqVis*** *(37)*, a program that meets the three criteria presented earlier. In the third section, we present some of the benefits of using ***SeqVis*** through a survey of an alignment of mitochondrial protein-coding genes from 53 taxa, which was recently used to estimate the evolutionary history of animals *(90)*. In the final section, we list some practical notes regarding ***SeqVis***. However, first we need to describe how sequence-specific nucleotide frequencies can be visualized in three, two, and one dimensions.

1.4. Visualization of Nucleotide Content

A genome, or a subsection thereof, may be regarded as a four-state information system, where (*i*) each state corresponds to a nucleotide, and (*ii*) the nucleotide frequencies (f_A, f_C, f_G, and f_T) meet the following conditions: $0 \leq f_A, f_C, f_G, f_T \leq 1$ and $f_A + f_C + f_G + f_T = 1$.

There are 13 ways to recode a sequence of nucleotides (i.e., *AGY*, *CTR*, *MGT*, *KAC*, *CGW*, *SAT*, *RY*, *KM*, *SW*, *AB*, *CD*, *GH*, and *TV*, where *R* = *A* or *G*, *Y* = *C* or *T*, *M* = *A* or *C*, *K* = *G* or *T*, *S* = *G* or *C*, *W* = *A* or *T*, *B* = *C* or *G* or *T*, *D* = *A* or *G* or *T*, *H* = *A* or *C* or *T*, and *V* = *A* or *C* or *G*), implying that DNA also may be regarded as three- and two-state information systems (conditions similar to those described in the previous paragraph are also met by these information systems).

The four-state information system has been used in a large number of phylogenetic studies, whereas the three- and two-state information systems have been used in only a few cases *(86, 91–95)*.

Visualizing the nucleotide content of a piece of DNA for each of these information systems is easy because the second condition dictates that $f_A + f_C + f_G + f_T = 1$. Therefore, it is possible to plot the nucleotide frequencies in one, two, and three dimensions, with the one-, two-, and three-dimensional plots representing, respectively, the two-, three-, and four-state information systems. In the one-dimensional plot, the two frequencies correspond to the distances from the two end points of a line with a length equal to 1 (**Fig. 4.2A**).

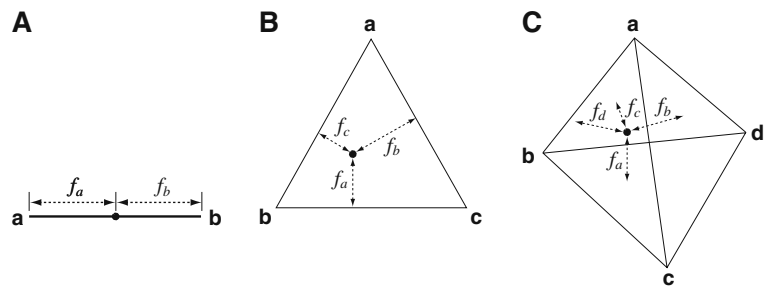


Fig. 4.2. Three ways of visualizing compositional heterogeneity: a linear plot (**A**), a de Finetti plot (**B**), and a tetrahedral plot (**C**). Each point in these plots corresponds to a unique observation in the two-, three-, and four-state information systems, while f_i represents the frequency of the *i*-th state in the corresponding information system.

In the two-dimensional plot, the three frequencies correspond to the distances between a point inside an equilateral triangle and the edges of this triangle (**Fig. 4.2B**), and in the three-dimensional plot, the four frequencies correspond to the distances between a point inside the regular tetrahedron and the sides of this tetrahedron (**Fig. 4.2C**). Each point in the linear plot (**Fig. 4.2A**) corresponds to a unique pair of nucleotide frequencies. Each point in the triangular plot (**Fig. 4.2B**), better known as the de Finetti plot *(96)*, corresponds to a unique triplet of nucleotide frequencies, and each point in the tetrahedral plot (**Fig. 4.2C**) corresponds to a unique quartet of nucleotide frequencies.

Given that a point in the linear plot, the de Finetti plot, and the tetrahedral plot corresponds to a unique nucleotide composition within these two-, three- and four-state information systems, it is clear that the plots offer many opportunities for detecting compositional heterogeneity in alignments of nucleotide sequences: the more scattered the points are within these plots, the more compositionally heterogeneous the sequences are. It is this realization that prompted the development of ***SeqVis*** *(37)*.

2. Program Usage

2.1. Introduction to SeqVis

SeqVis *(37)* is a stand-alone Java application that facilitates visual and statistical analyses of compositional heterogeneity in alignments of nucleotide sequences. The nucleotide content of each sequence may be displayed in a tetrahedral plot, a de Finetti plot, or a linear plot, and the compositional heterogeneity among the sequences may be analyzed using various exploratory and statistical methods, including the matched-pairs test of symmetry *(36)*.

In order to run, ***SeqVis*** requires the Java 3D package and Java Runtime Environment (version 5.0 or later). ***SeqVis*** is freely available from http://www.bio.usyd.edu.au/jermiin/SeqVis/ and can run on platforms that support Java. The features of ***SeqVis*** are outlined below.

2.2. The Graphical User Interface

The graphical user interface (GUI) of ***SeqVis*** is divided into four components (**Fig. 4.3**). The **Menu Bar**, which is found below the **Title Bar** in the GUI, provides access to all functions of ***SeqVis***. Below the **Menu Bar** is a **Tool Bar** with shortcut-buttons to commonly used actions, including (from left to right) the following:

- Open a file;
- Save a new file;
- Capture the plot displayed on the **Canvas**, which is found below the **Tool Bar**;

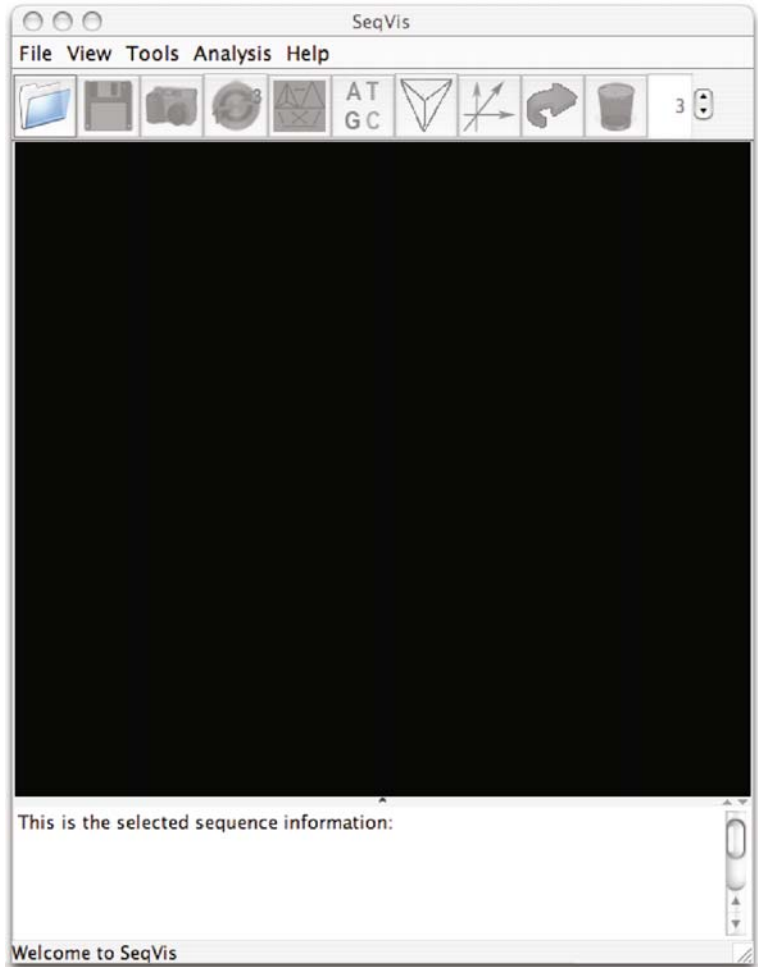


Fig. 4.3. The graphical user interface of ***SeqVis*** comprises a **Title Bar** (at the top) **Menu Bar** (below the **Title Bar**), a **Tool Bar** (below the **Menu Bar**), a **Canvas** (below the **Tool Bar**), and a **Data Panel** (below the **Canvas**), and a **Status Bar** (at the bottom). On the right-hand side of the **Data Panel** is a **Scroll Box**, which is located within a **Scroll Bar**, and two **Scroll Arrows**. The **Title Bar** has a **Close Button**, a **Maximize Button**, and a **Minimize Button**. The GUI can be resized: press the mouse over the lower right corner of the interface and drag it to the desired size. This is the first image encountered by users of ***SeqVis***.

- Toggle between displaying the nucleotide content in a single plot or in three plots—the latter is included to cater for protein-coding nucleotide sequences;
- Toggle between producing a tetrahedral plot, a de Finetti plot or a linear plot;
- Toggle between showing or hiding the labels on the plot;
- Toggle between showing or hiding the wireframe on the plot;
- Toggle between showing or hiding the axes on the plot;

- Toggle between viewing the plot while it is steady or rotating on a predefined axis;
- Remove individual data points from the plot;
- Change the size of data points in the plot.

The **Canvas**, which is found below the **Tool Bar**, is where the nucleotide content is displayed graphically. Transformations of the plots, which facilitate further examination of the scatter of data points, include rotation, translation, and zoom. The **Data Panel**, which is located below the **Canvas**, displays details pertaining to data points selected from the **Canvas**—the details provided include the following: sequence name, sequence length, frequencies of *A*, *T*, *G*, and *C* across all sites (or codon sites), and a count of other nucleotides.

2.3. File Handling

SeqVis is capable of opening and processing input files that have been saved as text-only files (*see* **Note 1**). Currently, ***SeqVis*** supports two types of data: sequence data and tab-delimited data. Sequence data must be saved in the sequential PHYLIP, NEXUS, and FASTA formats, and input files are assumed to use one of three indicative filename extensions: .phy, .fasta, and .nex (*see* **Note 2**).

Once the input file is open, ***SeqVis*** calculates the nucleotide content from raw sequence data, and additional sequences may be appended to the data using the **Add File** command from the **File** menu, which is located in the **Menu Bar**. For example, if the first input file contained an alignment of a gene from species A to F, then loading a second input file with data from species G to L will result in species A–L being displayed in the plot. A file with the revised data set may be saved in the sequential PHYLIP, NEXUS, and FASTA formats, and the plot, displayed on the **Canvas**, may be saved in the PNG or JPEG image file formats; the latter is achieved using the **Capture** command from the **File** menu (*see* **Note 3**).

SeqVis can also open tab-delimited input files; each line of the input file should contain the name and nucleotide content of a sequence (*see* **Note 2**). The advantage of this feature is that it is possible to display compositional heterogeneity presented in previously published tables (for an example, *see* **Fig. 4.4**).

2.4. Visualization

Having opened the input file, it may be convenient to change the appearance of the plot. For example, it may be necessary to change the colors of the **Canvas**, the wireframe and axes of the plot, and the data points. To change the color of the **Canvas**, use the **Color of Canvas** command from the **View** menu, which is found in the **Menu Bar**. Likewise, to change the colors of the wireframe or axes of the plot or the data points, use the **Color of Wireframe and Axes** or **Color of Points** commands, respectively, from the **View** menu. **Figure 4.5** shows the GUI, with (*i*) the color of the **Canvas**

```
Homo sapiens:                0.329     0.408     0.075     0.189
Pan paniscus:                0.368     0.368     0.039     0.224
Gorilla gorilla:             0.355     0.404     0.057     0.184
Hylobates syndactylus:       0.364     0.364     0.053     0.219
Papio anubis:                0.399     0.364     0.066     0.171
Papio hamadryas:             0.404     0.368     0.061     0.167
Theropithecus gelada:        0.408     0.320     0.057     0.215
Cerocebus galeritus:         0.439     0.338     0.022     0.202
Mandrillus leucophaeus:      0.417     0.338     0.048     0.197
Macaca fascicularis:         0.408     0.368     0.061     0.162
Macaca mulatta:              0.399     0.333     0.057     0.211
Cercopithecus aethiops:      0.395     0.351     0.057     0.197
Lagothrix lagothrica:        0.405     0.336     0.043     0.216
Alouatta paliatta:           0.388     0.306     0.060     0.246
Tarsiers syrichta:           0.430     0.285     0.026     0.259
Tarsiers bancanus:           0.412     0.232     0.035     0.320
Daubentonia madagascarensis: 0.382     0.333     0.061     0.224
Hapalemur griseus:           0.434     0.224     0.040     0.303
Lemur catta:                 0.443     0.250     0.022     0.285
Lemur macaco macaco:         0.408     0.232     0.057     0.303
Varecia variegata:           0.425     0.259     0.035     0.281
Cheirogaleus medius:         0.412     0.272     0.044     0.272
Propithecus tattersalli:     0.373     0.303     0.088     0.237
Nycticebus coucang:          0.377     0.289     0.075     0.259
Galago senegalensis:         0.404     0.351     0.053     0.193
```

Fig. 4.4. ***SeqVis*** is capable of reading input files with tab-delimited data. The frequencies ($0 \leq f_A, f_C, f_G, f_T \leq 1$) in each row must add up to one. The data presented here were obtained from the first and last four columns of Table 3 in Adkins and Honeycutt *(97)*.

set to white, (*ii*) the color of the wireframe and data points set to black, and (*iii*) the data points enlarged to 7 (done via the **Tool Bar**).

Apart from changing the colors of critical components of a plot, it is also possible to hide the wireframe of a plot and show axes of the plot, either jointly or individually. To do this, use the **Toggle Wireframe** and **Toggle Axes** commands from the **View** menu. The advantage offered by using the wireframe is that it is easier to see how data points are dispersed (**Fig. 4.6A**), whereas the advantage offered by using the axes is that these allow the user to determine where data points are located within the space, especially in relation to the centre of plot (**Fig. 4.6B**), which corresponds to a uniform nucleotide content.

SeqVis also allows transformations of a plot on the **Canvas**. The easiest way to rotate a plot is to use the **Auto Rotate** command from the **View** menu, or use the shortcut (from the **Tool Bar**). If the plot must be viewed from other angles, then a platform-specific combination of mouse and key actions must be used (*see* **Note 4**). The advantage of rotating a plot is that it facilitates identifying common trends in the scatter of data points. Translations, enlargements, or reductions of a plot on the **Canvas** are also achieved using platform-specific combinations of mouse and key actions (*see* **Note 4**). The advantage of using these transformations is that it becomes easier to draw attention to important details in the plot before it is saved using the **Capture** command (for details,

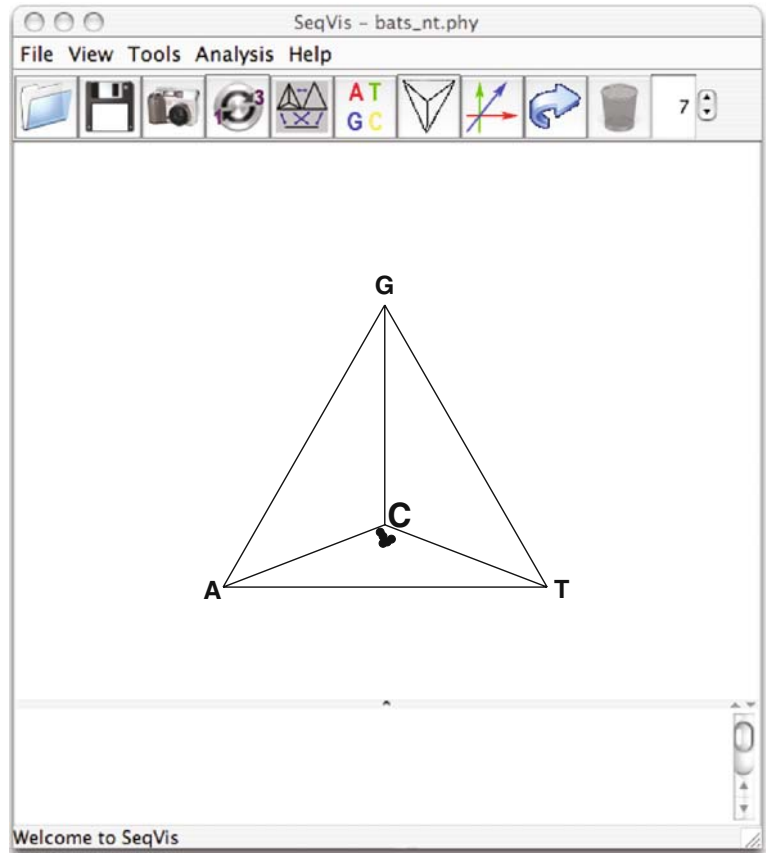


Fig. 4.5. The graphical user interface of ***SeqVis*** with the colors of the wireframe and data points set to black and the color of the **Canvas** set to white. The tetrahedral plot shows the compositional heterogeneity among the sequences shown in **Fig. 4.1**.

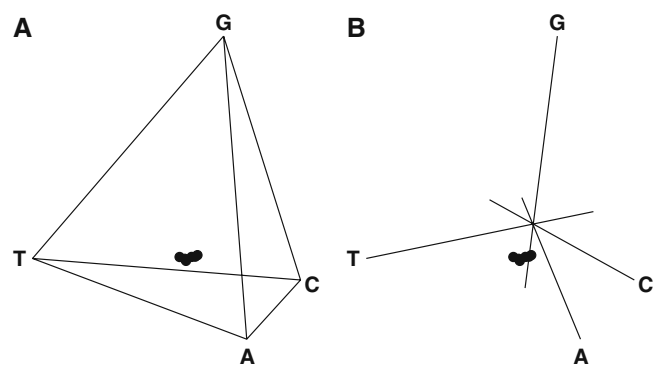


Fig. 4.6. A tetrahedral plot with the wireframe (**A**) or axes (**B**) displayed. The plot was rotated to show the scatter of data points from another angle. In this example, it can be concluded that there is compositional heterogeneity among the sequences and that $f_G < 0.25$. The sequences are those shown in **Fig. 4.1**.

see above). Furthermore, there are four level planes, each of which is parallel to a surface of the tetrahedron. The level planes can be moved closer to or further away from the opposite corner of a tetrahedron by using the up (▲) and down (▼) keys on the keyboard, whereas the level of transparency of the level planes is controlled by the left (◀) or right (▶) keys on the keyboard. The level planes can be used to identify more precisely where data points are located in the tetrahedral plot.

SeqVis also allows users to survey the nucleotide content at individual codon sites of protein-coding genes: use the **Select Sites** command from the **View** menu, or use the shortcut (from the **Tool Bar**). In the latter case, three plots will be shown side-by-side on the **Canvas**, with the plot on the left showing the nucleotide content at first codon sites, the plot in the middle showing the nucleotide content at second codon sites, and the plot on the right showing the nucleotide content at third codon sites (**Fig. 4.7**). The plots may be transformed using platform-specific combinations of mouse and key actions (*see* **Note 4**).

Alternatively, when the plots need to be viewed from the same angle use the **All Sites**, **1st Codon Site**, **2nd Codon Site**, or **3rd Codon Site** commands from the **Select Sites** command, from the **View** menu (**Fig. 4.8**).

To further assist the visual exploration of sequence data, ***SeqVis*** allows users to search for a specific sequence using either its name or sequence; use the **Search by Name** or **Search by Sequence** commands from the **Tools** menu, which is found in the **Menu Bar**. Alternatively, it is possible to obtain information about a particular data point of interest (*see* **Note 5**).

The details and visibility of the sequences can be updated via the control panel invoked using the **Display Sequence Details** command from the **Tools** menu. It is possible to remove data points from the plot: use the **Display Sequence Details** command from the **Tools** menu, and use the mouse to deselect the sequences to be removed from the plot.

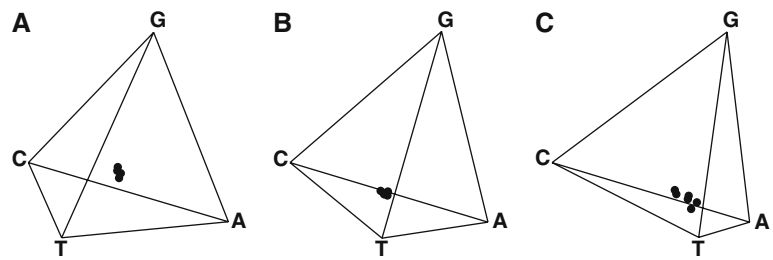


Fig. 4.7. Tetrahedral plots for first codon sites (**A**), second codon sites (**B**), and third codon sites (**C**). The plots were obtained using the **Select Sites** shortcut from the **Tool Bar**. In this example, it can be concluded that the compositional heterogeneity is most prevalent at third codon sites and least prevalent at second codon sites. The data are those shown in **Fig. 4.1**.

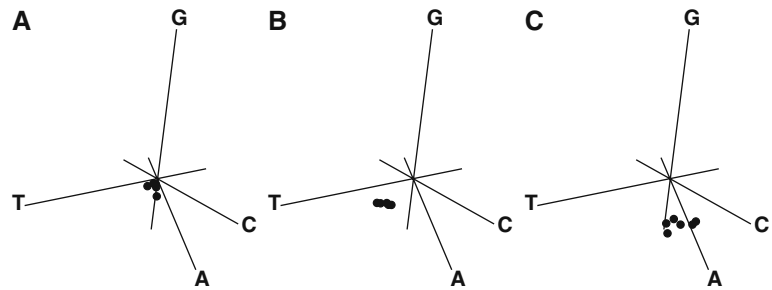


Fig. 4.8. Tetrahedral plots for first codon sites (**A**), second codon sites (**B**), and third codon sites (**C**). The plots were obtained using the **Select Sites** command from the **View** menu. In this example, it can be concluded that the nucleotide content varies among the codon sites, which implies that it would be inappropriate to assume that the codon sites evolved under the same conditions. Moreover, it can be concluded that the nucleotide content is almost uniform at first codon sites and that *T* and *A* are the most abundant nucleotides at, respectively, second and third codon sites. The data are those shown in **Fig. 4.1**.

SeqVis also allows users to investigate the effect of recoding the nucleotides. Recoding is a procedure in which composite sets of nucleotides are represented by a single letter (*see* **Note 6**). ***SeqVis*** can calculate the nucleotide content of recoded sequences and display the results in either a de Finetti plot (**Fig. 4.9**) or a linear plot (**Fig. 4.10**) (*see* **Note 7**). Collectively, these two plots are called lower-dimensional plots (*see* **Note 8**).

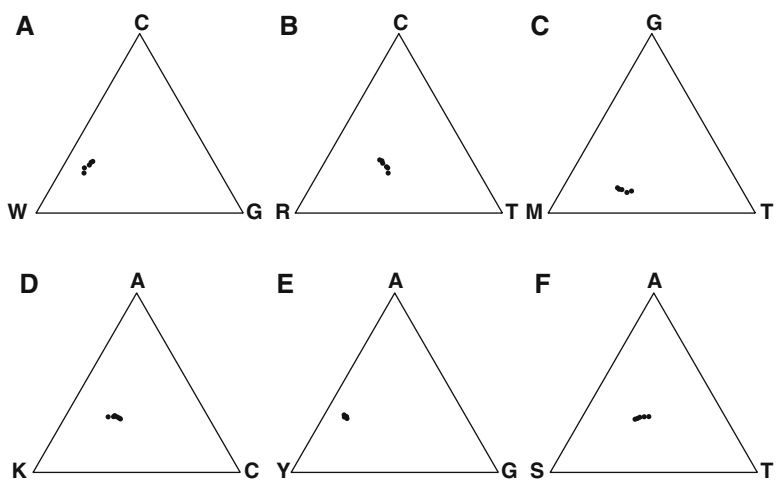


Fig. 4.9. Six de Finetti plots based on third codon sites from the alignment in **Fig. 4.1**. The following three-state information systems used: *WCG* (**A**), *RCT* (**B**), *MGT* (**C**), *KAC* (**D**), *YAG* (**E**), and *SAT* (**F**). In this example, the least compositionally heterogeneous alignment of third codon sites is obtained using the YAG information system. If time-reversible Markov models were to be used in a phylogenetic analysis of these data, it would be prudent to use this information system for third codon sites. Gibson et al. *(95)* used this type of recoding in their study of the evolution of mammalian mitochondrial genomes.

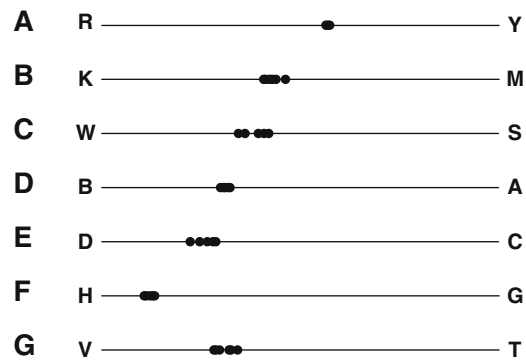


Fig. 4.10. Seven linear plots based on third codon sites from the alignment in **Fig. 4.1**. The following two-state information systems used: *RY* (**A**), *KM* (**B**), *WS* (**C**), *BA* (**D**), *DC* (**E**), *HG* (**F**), and *VT* (**G**). In this example, the most compositionally homogeneous alignment of third codon sites is obtained using the RY information system. If time-reversible Markov models were to be used in a phylogenetic analysis of these data, it would be prudent to use this information system for third codon sites. RY-recoded nucleotides have been used several times, including in phylogenetic studies of RNA genes *(91)*, chloroplast and nuclear genes *(92),* and metazoan mitochondrial genes *(86, 93, 94)*.

2.5. Analysis

To complement the visualization, ***SeqVis*** also provides access to analytical methods that may help users gain a better understanding of the nature and extent of compositional heterogeneity among the sequences. Five such methods may be accessed from the **Analysis** menu, which is located in the **Menu Bar**. Conceptually, the five methods are partitioned into two categories, with two methods for analyzing the extent of compositional heterogeneity, and three methods for clustering the data according to their nucleotide content. Methods belonging to the first category compare the sequences in a heterogeneity analysis or using the matched-pairs test of symmetry *(36)*, whereas methods belonging to the second category divide the sequences into subsets using *k*-mean clustering, self-organizing clustering, and hierarchical clustering.

Results obtained from a cluster analysis may become useful when assessing the reliability of an inferred phylogeny. For example, if the phylogeny is discovered to arrange the sequences in a manner similar to how they cluster simply on the basis of their nucleotide content, then there is reason for concern: the phylogeny may depict the compositional differences rather than the evolutionary history of the data.

Details pertaining to the five methods are outlined in the following five subsections.

2.5.1. Analysis of Heterogeneity

Occasionally, it may be useful to color-code the data points in a manner that reflects how far they are located from their centroid. To do this, use the **Analysis of Heterogeneity** command from the **Analysis** menu. The clustering of data is done using

three-dimensional Euclidean distances between data points in the tetrahedron, and data points are colored according to their distances from the centroid. Data points located close to the centroid are blue and those located further away are displayed in a range of other colors; hence, if the data points are shown in more than one color, the sequences are likely to be compositionally heterogeneous, and it would be prudent to do the matched-pairs test of symmetry before using the sequences in a phylogenetic study.

Sometimes, the removal of outliers may be sufficient to reduce compositional heterogeneity. It should be noted, however, that the analysis of heterogeneity only serves as a rough guide to help identify evidence of compositional heterogeneity; it does not replace the need to use the matched-pairs test of symmetry. Indeed, even if all the data points are blue, we recommend that the matched-pairs test of symmetry be used.

2.5.2. Matched-Pairs Test of Symmetry

As stated earlier in this chapter, compositional heterogeneity among aligned sequences of nucleotides implies that the data cannot have evolved under stationary, reversible, and homogeneous conditions. To corroborate this implication, ***SeqVis*** offers access to a suitable statistical test *(36)*: use the **Matched-pairs Test of Symmetry** command from the **Analysis** menu. The test compares the off-diagonal elements of a divergence array (i.e., d_{ij} and d_{ji}, for $i \neq j$) derived from pairs of sequences, and produces a test statistic, S_B^2, that is asymptotically distributed as a χ^2-variate on $v = n(n-1)/2$ degrees of freedom (n is the number of states in the information system).

Under the null hypothesis (i.e., that the diverging sequences have evolved under the same stationary, reversible, and homogeneous conditions), the distribution of p-values obtained from the tests is uniform on (0, 1) and unaffected by sequence divergence and invariant sites, so it is straightforward to draw a sound conclusion from the tests. For example, if an alignment produces a non-uniform distribution of predominantly small p-values, it can be concluded that the sequences have evolved under (*i*) non-stationary, non-reversible, and non-homogeneous conditions or (*ii*) stationary, non-reversible, and non-homogeneous conditions. Compositional heterogeneity indicates evolution under the former set of conditions but not the latter *(40)*. **Figure 4.11** shows the results obtained from using the matched-pairs test of symmetry.

2.5.3. k*-mean Clustering*

Three data clustering methods are implemented in ***SeqVis***. The simplest method of clustering the sequences according to their nucleotide content is obtained through k-mean clustering: use the ***k*-mean Clustering** command from the **Analysis** menu, and enter the value of k (i.e., the number of clusters). The clustering of data points is done using the following algorithm (using Euclidean distances, in three dimensions, between data points in the tetrahedron):

```
****  Matched-pairs Test of Symmetry ****

Each number shows the probability of the sequences having
evolved under the same time-reversible conditions

Sequence No.  0        1        2        3        4        5
Ajam 0
Ctub 1        0.1946
Mtub 2        0.6202   0.6985
Pabr 3        0.4306   0.2374   0.7925
Rmon 4        0.0525   0.1746   0.2730   0.1650
Rpum 5        0.1682   0.3225   0.5434   0.1643   0.2998
              0        1        2        3        4        5


____ Summary of Matched-pairs Tests of Symmetry ____
                          Number  Proportion
P-values < 0.05:             0     (0.0000)
P-values < 0.01:             0     (0.0000)
P-values < 0.005:            0     (0.0000)
P-values < 0.001:            0     (0.0000)
P-values < 0.0005:           0     (0.0000)
P-values < 0.0001:           0     (0.0000)
P-values < 0.00005:          0     (0.0000)
Number of test:             15
Smallest p-value:           0.0525
_____________________________________________________
```

Fig. 4.11. Results from matched-pairs tests of symmetry based on third codon sites from the alignment in **Fig. 4.1**. The *upper* half of the figure shows the *p*-values for all pairs of sequences, whereas the *lower* half of the figure shows a summary of the distribution of *p*-values, the number of tests conducted, and the smallest *p*-value obtained. In this example, none of the tests produced a *p*-value < 0.05, implying that there is no statistical support for rejecting the null hypothesis of evolution under stationary, reversible, and homogeneous conditions.

1. Initiate k clusters and randomly select k points to be the centroid of these clusters;
2. Assign each data point to the nearest cluster centroid;
3. Re-calculate the coordinates of the centroid of each cluster based on the coordinates of all the data points in that cluster;
4. Repeat 2–3 until all clusters are stable.

The quality of the result depends on the value of k. One should choose k to be the most likely number of clusters present in the data based on an inspection of the tetrahedron plot. Selecting too high a value of k (i.e., one not supported by the data) may lead to a meaningless result.

2.5.4. Self-Organizing Clustering

To alleviate the problem of having to select the optimal value of k, one may use the **Self-Organizing Clustering** command from the **Analysis** menu. The algorithm for self-organizing clustering *(98)* is similar to that used for k-mean clustering, except for two major differences: (*i*) initially, in Step 1 k equals the number of data

points, and (*ii*) the closest pair of clusters are merged into a single cluster after Step 3 if their centroids are closer to one another than a predefined Euclidean distance.

2.5.5. Hierarchical Clustering

Finally, it is possible to use ***SeqVis*** to arrange the sequences in a binary tree according to their nucleotide content: use the **Hierarchical Clustering** command from the **Analysis** menu. This method deviates from the other clustering methods in two important ways: (*i*) the nucleotide frequencies are used as coordinates for the data points, and (*ii*) the similarities between these are summarized in the form of a binary tree rather than as a set of non-overlapping clusters. The hierarchical clustering algorithm operates as follows:

1. Create and assign a unique node to each data point, and place the nodes in an array that is called NArray. The position of each node is assigned to be the coordinates of the data point it contains (i.e., f_A, f_C, f_G, f_T);
2. While NArray contains more than one node
   a. Using the Euclidean distance between pairs of nodes, find the pair of nodes, *a* and *b*, that are closest to one another;
   b. Create a new node, *c*, and assign to it the coordinates of the midpoint of a line that connects *a* and *b*;
   c. Create two edges, *ac* and *bc*, and assign to each of these edges a length, l_{ac} and l_{bc}, that is equal to half of the Euclidean distance between *a* and *b*;
   d. Remove node *a* and *b* from NArray, and add *c* into NArray;

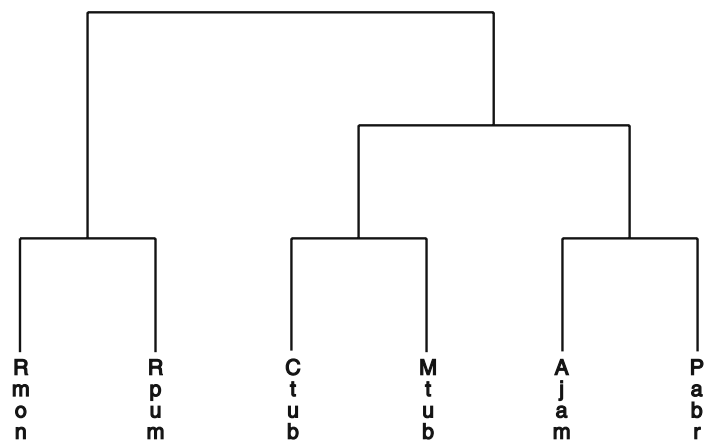


Fig. 4.12. Result from a hierarchical clustering based on third codon sites from the alignment in **Fig. 4.1**. According to the tree, the nucleotide composition of (*i*) *Rhinolophus monoceros* is similar to that of *R. pumilus*; (*ii*) *Artibeus jamaicensis* is similar to that of *Pipistrellus abramus*; and (*iii*) *Mystacina tuberculata* is similar to that of *Chalinolobus tuberculatus*.

3. Let the order in which the nodes were removed from NArray defines a binary tree, and let l_{ij} be the distance separating nodes i and j, where $i \neq j$.

The above algorithm is similar to the standard hierarchical clustering algorithm using average linkage. In ***SeqVis***, the resulting tree is visualized in a separate window (**Fig. 4.12**).

3. An Example

The evolutionary relationship among representatives of extant animal phyla is a hotly debated issue with a growing number of molecular phylogenetic studies having produced a variety of interesting, but also incongruent, evolutionary hypotheses *(90, 99–110)*. Very long alignments of concatenated genes or gene products of nuclear or mitochondrial origin were used in many of these studies, and the reasons given for the inability of these studies to reach an agreement include that the data (*i*) were obtained from different sets of species; (*ii*) comprised different sets of genes or gene products; and (*iii*) were analyzed differently, with special attention paid to different factors known to bias phylogenetic results. However, in only one of these studies was compositional heterogeneity recognized as a possible problem (i.e., ref. *105*), despite the abovementioned body of evidence suggesting that compositional heterogeneity might result in biased phylogenetic estimates if these were to be obtained under stationary, reversible, and homogeneous conditions. Most of the phylogenetic studies cited above focused on amino acid sequences but a few also used nucleotide sequences. To demonstrate the usefulness of ***SeqVis*** in the context of a phylogenetic study, we obtained an alignment of protein-coding mitochondrial genes from 53 species of animals. Recently, the alignment was used to estimate the evolutionary history of animals *(90)*.

Figure 4.13A presents the tetrahedral plot for all sites in the alignment, and **Fig. 4.13B–D** present the tetrahedral plots from the alignment's first, second, and third codon sites. Clearly, there is compositional heterogeneity in the alignment, and it appears to be most conspicuous at third codon sites (**Fig. 4.13D**) and least conspicuous at second codon sites (**Fig. 4.13C**). The implication of these plots is that the codon sites are unlikely to have evolved under the same stationary, reversible, and homogeneous conditions.

To corroborate whether this is the case, the matched-pairs test of symmetry was used in conjunction with first, second, and third codon sites of the alignment. **Table 4.2** summarizes the distribution of *p*-values for each codon site. The distributions of *p*-values

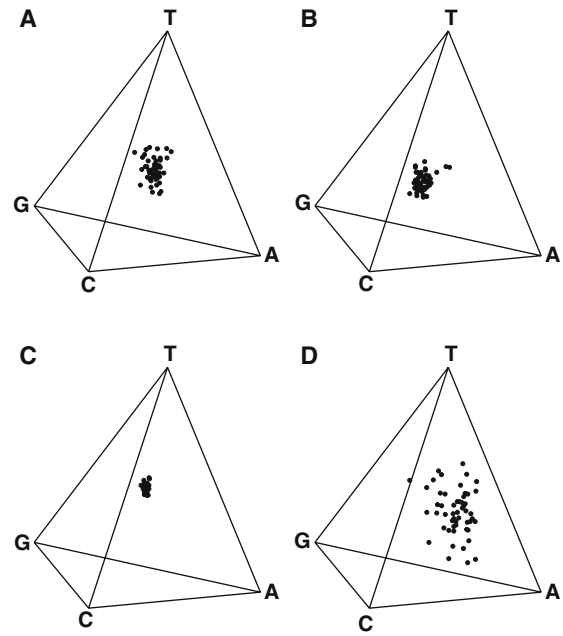


Fig. 4.13. Tetrahedral plots for: all codon sites (**A**), first codon sites (**B**), second codon sites (**C**), and third codon sites (**D**). The plots were obtained using the **Select Sites** command from the **View** menu, and were based on an alignment of protein-coding mitochondrial genes used to infer the evolutionary relationship among 53 species of animals *(90)*. Based on these plots, it can be concluded that the nucleotide content at all codon sites varies greatly among the sequences, especially at third codon sites, implying that it would be unwise to assume that the genomes have evolved under stationary, reversible, and homogeneous conditions.

Table 4.2
Summary of results from matched-pairs tests of symmetry[a]

	First codon sites		Second codon sites		Third codon sites	
Threshold	**Number[b]**	**Proportion**	**Number[b]**	**Proportion**	**Number[b]**	**Proportion**
0.05	1306	0.948	1061	0.770	1364	0.990
0.01	1259	0.914	919	0.667	1349	0.979
0.005	1233	0.895	859	0.623	1340	0.972
0.001	1182	0.858	762	0.553	1331	0.966
0.0005	1158	0.840	728	0.528	1327	0.963
0.0001	1114	0.808	661	0.480	1316	0.955
0.00005	1093	0.793	633	0.460	1307	0.948

[a] The alignment of protein-coding mitochondrial genes from 53 animals was obtained from Bourlat et al. *(90)*.
[b] The number of times that the matched-pairs test of symmetry resulted in a *p*-value below the threshold (number of tests: 1378).

clearly show that the evolutionary processes are unlikely to have been stationary, reversible, and homogeneous at these codon sites, implying that it would be unwise to analyze these data using a phylogenetic approach that assumes a stationary, reversible, and homogeneous evolutionary process.

We then recoded the nucleotides in an attempt to reduce the compositional heterogeneity, but none of the 13 ways of recoding nucleotides resulted in a compositionally homogeneous alignment.

Given these results, one might wish to know how these data should have been analyzed. One solution might be to analyze the corresponding alignment of amino acids; however, given that compositional heterogeneity at the first and second codon sites translates into compositionally heterogeneity in the corresponding alignment of amino acids *(3)*, it would be necessary to use phylogenetic methods that can account for the more general mode of evolution. Alternatively, one might analyze the alignment of nucleotides using phylogenetic methods that can account for more general modes of molecular evolution. In addition, it may also be necessary removing some of the offending sequences and/or sites from the alignment. Having done this, recoding of the nucleotides or amino acids *(87)* is yet another possibility to consider. Obviously, the recoded and reduced data set would have to be tested again using the matched-pairs test of symmetry.

4. Notes

1. Before opening an input file, it is best to check that it conforms to the file format expected by ***SeqVis***. Different operating systems define the line break differently, so to obtain an input file that can be processed using ***SeqVis***, we recommend that the input file be (*i*) prepared using a standard text editor (e.g., ***TextWrangler*** or ***BBEdit***, both products of Bare Bones Software) and (*ii*) saved in the text-only format.
2. Before opening an input file, it is best to ascertain that the data conforms to the formats assumed by ***SeqVis***. With respect to sequence data, note that there are different versions of the NEXUS and PHYLIP formats. With respect to tab-delimited data, ***SeqVis*** assumes that the frequencies add up to unity—failure to comply with this requirement might lead errors. If in doubt, open one of the sample input files provided with ***SeqVis*** and copy your data into it.

3. Because the **Capture** command saves the plots in the JPEG or PNG formats, ***SeqVis*** produces low-resolution plots. Hence, it may be desirable to improve the image quality before the plots are ready for publication. In fact, this was case for this chapter: first, we captured the plots in the JPEG format; then, we used ***Adobe Photoshop CS2*** to convert the JPEG-formatted file to a PDF-formatted file; finally, we opened the PDF-formatted file using ***Adobe Illustrator CS2***, retraced the plot, and saved the file in the publication-ready, PDF format. Most of the images presented here were modified in this manner.
4. Rotating, translating, and resizing the plot is done using platform-dependent commands. On computers running Mac OS X, the plot can be rotated by pressing the mouse while moving it over the plot, translated by pressing the Command key and the mouse while moving the latter over the plot, and resized by pressing the Option key and the mouse while moving the latter up or down over the plot. A pop-up menu with other commands is shown by pressing the Ctrl key and then the mouse. On computers running Microsoft Window™ or Linux, the plot can be rotated, translated, and resized by simply moving the mouse while pressing the left, right, and middle buttons on the mouse. A pop-up menu with other commands is shown by pressing the Ctrl key and then the right button on the mouse (please note that the function of the left and right buttons depends on how the mouse is configured).
5. To display details about a particular sequence in the **Data Panel**, it is necessary first to select the data point of interest (e.g., an outlier): use the mouse to click on the data point of interest (on the **Canvas**).
6. Going from the classical four-state information system (i.e., *A*, *C*, *G*, and *T*) to the three- or two-state information system often leads to a loss of information. At an initial glance, the loss might not be desirable, but if recoding of nucleotides reduces the compositional heterogeneity among sequences, then it might in fact raise the likelihood of inferring the correct phylogeny.
7. When inspecting sequences using lower-dimensional information systems, it is best not to rotate the tetrahedron before switching to the lower-dimensional information systems; provided the initial tetrahedral plot has not been modified, ***SeqVis*** is designed to present the de Finetti plot and the linear plot flatly on the **Canvas**.
8. Sometimes it is useful to visualize how much individual nucleotide frequencies vary across the sequences. To do this, recode the nucleotides to *AB*, *CD*, *GH*, and *TV*, and display the plots. For example, based on **Fig. 4.10D–G**, it may be

concluded that compositional heterogeneity at third codon site of the sequences in **Fig. 4.1** is due largely to variation in the frequencies of *C* and *T*.

Acknowledgment

This research was partly funded by Discovery Grants from the Australian Research Council to LSJ and an Australian Postgraduate Award and a NICTA Research Project Award to JWKH.

References

1. Jukes, T. H., and Bhushan, V. (1986) Silent nucleotide substitutions and G+C content of some mitochondrial and bacterial genes. *J Mol Evol* **24,** 39–44.
2. Jermiin, L. S., Graur, D., Lowe, R. M., and Crozier, R. H. (1994) Analysis of directional mutation pressure and nucleotide content in mitochondrial cytochrome *b* genes. *J Mol Evol* **39,** 160–73.
3. Foster, P. G., Jermiin, L. S., and Hickey, D. A. (1997) Nucleotide composition bias affects amino acid content in proteins coded by animal mitochondria. *J Mol Evol* **44,** 282–88.
4. IHGSC (2001) Initial sequencing and analysis of the human genome. *Nature* **409,** 860–921.
5. Lee, K. Y., Wahl, R., and Barbu, E. (1956) Contenu en bases puriques et pyrimidiques des acides desoxyribonucleiques des bacteries. *Annales de l'Institut Pasteur* **91,** 212–24.
6. Belozersky, A. N., and Spirin, A. S. (1958) A correlation between the compositions of deoxyribonucleic and ribonucleic acids. *Nature* **182,** 111–12.
7. Rolfe, R., and Meselson, M. (1959) The relative homogeneity of microbial DNA. *Proc Natl Acad Sci USA* **45,** 1039–42.
8. Sueoka, N. (1959) A statistical analysis of deoxyribonucleic acid distribution in density gradient centrifugation. *Proc Natl Acad Sci USA* **45,** 1480–90.
9. Sueoka, N., Marmur, J., and Doty, P. (1959) Dependence of the density of deoxyribonucleic acids on guanine-cytosine content. *Nature* **183,** 1429–31.
10. Crozier, R. H., and Crozier, Y. C. (1993) The mitochondrial genome of the honeybee *Apis mellifera*: complete sequence and genome organization. *Genetics* **133,** 97–117.
11. Jermiin, L. S., Graur, D., and Crozier, R. H. (1995) Evidence from analysis of intergenic regions for strand-specific directional mutation pressure in metazoan mitochondrial DNA. *Mol Biol Evol* **12,** 558–63.
12. Bernardi, G., Olofsson, B., Filipski, J., Zerial, M., Salinas, J., Cuny, G., Meunier-Rotival, M., and Rodier, F. (1985) The mosaic genome of warm-blooded vertebrates. *Science* **228,** 953–58.
13. Mouchiroud, D., D'Onofrio, G., Aïssani, B., Macaya, G., Gautier, C., and Bernardi, G. (1991) The distribution of genes in the human genome. *Gene* **100,** 181–87.
14. Bernardi, G. (2004) *Structural and Evolutionary Genomics: Natural Selection in Genome Evolution*, Elsevier, Amsterdam.
15. Sharp, P. M., and Lloyd, A. T. (1993) Regional base composition variation along yeast chomosome III: evolution of chromosome primary structure. *Nucleic Acids Res* **21,** 179–83.
16. Bradnam, K. R., Seoighe, C., Sharp, P. M., and Wolfe, K. H. (1999) G+C content variation along and among *Saccharomyces cerevisiae* chromosomes. *Mol Biol Evol* **16,** 666–75.
17. Hori, H., and Osawa, S. (1987) Origin and evolution of organisms as deduced from 5S ribosomal RNA sequences. *Mol Biol Evol* **4,** 445–72.
18. Weisburg, W. G., Giovannoni, S. J., and Woese, C. R. (1989) The *Deinococcus - Thermus* phylum and the effect of ribosomal RNA composition on phylogenetic tree

construction. *Syst Appl Microbiol* **11,** 128–34.

19. Lockhart, P. J., Howe, C. J., Bryant, D. A., Beanland, T. J., and Larkum, A. W. D. (1992) Substitutional bias confounds inference of cyanelle origins from sequence data. *J Mol Evol* **34,** 153–62.
20. Lockhart, P. J., Penny, D., Hendy, M. D., Howe, C. J., Beanland, T. J., and Larkum, A. W. D. (1992) Controversy on chloroplast origins. *FEBS Lett* **301,** 127–31.
21. Hasegawa, M., and Hashimoto, T. (1993) Ribosomal RNA trees misleading? *Nature* **361,** 23.
22. Olsen, G. J., and Woese, C. R. (1993) Ribosomal RNA: a key to phylogeny. *FASEB J* **7,** 113–23.
23. Sogin, M. L., Hinkle, G., and Leipe, D. D. (1993) Universal tree of life. *Nature* **362,** 795.
24. Forterre, P., Benachenhou-Lafha, N., and Lebadan, B. (1993) Universal tree of life. *Nature* **362,** 795.
25. Klenk, H. P., Palm, P., and Zillig, W. (1994) DNA-dependent RNA polymerases as phylogenetic marker molecules. *Syst Appl Microbiol* **16,** 638–47.
26. Pettigrew, J. D. (1994) Genomic evolution: flying DNA. *Curr Biol* **4,** 277–80.
27. Schultes, E., Hraber, P. T., and LaBean, T. H. (1997) Global similarities in nucleotide base composition among disparate functional classes of single-stranded RNA imply adaptive evolutionary convergence. *RNA* **3,** 792–806.
28. van den Bussche, R. A., Baker, R. J., Huelsenbeck, J. P., and Hillis, D. M. (1998) Base compositional bias and phylogenetic analyses: a test of the "flying DNA" hypothesis. *Mol Phylogenet Evol* **10,** 408–16.
29. Foster, P. G., and Hickey, D. A. (1999) Compositional bias may affect both DNA-based and protein-based phylogenetic reconstructions. *J Mol Evol* **48,** 284–90.
30. Chang, B. S. W., and Campbell, D. L. (2000) Bias in phylogenetic reconstruction of vertebrate rhodopsin sequences. *Mol Biol Evol* **17,** 1220–31.
31. Tarrío, R., Rodriguez-Trelles, F., and Ayala, F. J. (2001) Shared nucleotide composition biases among species and their impact on phylogenetic reconstructions of the Drosophilidae. *Mol Biol Evol* **18,** 1464–73.
32. Ho, S. Y. W., and Jermiin, L. S. (2004) Tracing the decay of the historical signal in biological sequence data. *Syst Biol* **53,** 623–37.
33. Goremykin, V. V., and Hellwig, F. H. (2005) Evidence for the most basal split in land plants dividing bryophyte and tracheophyte lineages. *Plant Systemat Evol* **254,** 93–103.
34. Murray, S., Fløe Jørgensen, M., Ho, S. Y. W., Patterson, D. J., and Jermiin, L. S. (2005) Improving the analysis of dinoflagelate phylogeny based on rDNA. *Protist* **156,** 269–86.
35. Jayaswal, V., Jermiin, L. S., and Robinson, J. (2005) Estimation of phylogeny using a general Markov model. *Evol Bioinf Online* **1,** 62–80.
36. Ababneh, F., Jermiin, L. S., Ma, C., and Robinson, J. (2006) Matched-pairs tests of homogeneity with applications to homologous nucleotide sequences. *Bioinformatics* **22,** 1225–31.
37. Ho, J. W. K., Adams, C. E., Lew, J. B., Matthews, T. J., Ng, C. C., Shahabi-Sirjani, A., Tan, L. H., Zhao, Y., Easteal, S., Wilson, S. R., and Jermiin, L. S. (2006) SeqVis: Visualization of compositional heterogeneity in large alignments of nucleotides. *Bioinformatics* **22,** 2162–63.
38. Jayaswal, V., Robinson, J., and Jermiin, L. S. (2007) Estimation of phylogeny and invariant sites under the General Markov model of nucleotide sequence evolution. *Syst Biol* **56,** 155–62.
39. Hyman, I. T., Ho, S. Y. W., and Jermiin, L. S. (2007) Molecular phylogeny of Australian Helicarionidae, Microcystidae and related groups (Gastropoda: Pulmonata: Stylommatophora) based on mitochondrial DNA. *Mol Phylogenet Evol* **45,** 792–812.
40. Jermiin, L. S., Jayaswal, V., Ababneh, F., and Robinson, J. (2008) *Bioinformatics: Data, Sequence Analysis, and Evolution* (Keith, J., Ed.), Vol. I, pp. 331–64, Humana Press, Totowa, NJ.
41. Bryant, D., Galtier, N., and Poursat, M.-A. (2005) Mathematics in Evolution and Phylogeny (Gascuel, O., Ed.), pp. 33–62, Oxford University Press, Oxford.
42. Ababneh, F., Jermiin, L. S., and Robinson, J. (2006) Generation of the exact distribution and simulation of matched nucleotide sequences on a phylogenetic tree. *J Math Model Algorithm* **5,** 291–308.
43. Kolmogoroff, A. (1936) Zur theorie der Markoffschen ketten. *Math Ann* **112,** 155–60.

44. Lockhart, P. J., Steel, M. A., Hendy, M. D., and Penny, D. (1994) Recovering evolutionary trees under a more realistic model of sequence evolution. *Mol Biol Evol* **11,** 605–12.
45. Conant, G. C., and Lewis, P. O. (2001) Effects of nucleotide composition bias on the success of the parsimony criterion on phylogenetic inference. *Mol Biol Evol* **18,** 1024–33.
46. Jermiin, L. S., Ho, S. Y. W., Ababneh, F., Robinson, J., and Larkum, A. D. W. (2004) The biasing effect of compositional heterogeneity on phylogenetic estimates may be underestimated. *Syst Biol* **53,** 638–43.
47. Gowri-Shankar, V., and Rattray, M. (2006) Compositional heterogeneity across sites: effects on phylogenetic inference and modelling the correlations between base frequencies and substitution rate. *Mol Biol Evol* **23,** 352–64.
48. Jukes, T. H., and Cantor, C. R. (1969) *Mammalian Protein Metabolism* (Munro, H. N., Ed.), pp. 21–132, Academic Press, New York.
49. Lanave, C., Preparata, G., Saccone, C., and Serio, G. (1984) A new method for calculating evolutionary substitution rates. *J Mol Evol* **20,** 86–93.
50. Barry, D., and Hartigan, J. A. (1987) Statistical analysis of hominoid molecular evolution. *Stat Sci* **2,** 191–210.
51. Reeves, J. (1992) Heterogeneity in the substitution process of amino acid sites of proteins coded for by the mitochondrial DNA. *J Mol Evol* **35,** 17–31.
52. Steel, M. A., Lockhart, P. J., and Penny, D. (1993) Confidence in evolutionary trees from biological sequence data. *Nature* **364,** 440–42.
53. Lake, J. A. (1994) Reconstructing evolutionary trees from DNA and protein sequences: paralinear distances. *Proc Natl Acad Sci USA* **91,** 1455–59.
54. Steel, M. A. (1994) Recovering a tree from the leaf colourations it generates under a Markov model. *Appl Math Lett* **7,** 19–23.
55. Galtier, N., and Gouy, M. (1995) Inferring phylogenies from DNA sequences of unequal base compositions. *Proc Natl Acad Sci USA* **92,** 11317–21.
56. Steel, M. A., Lockhart, P. J., and Penny, D. (1995) A frequency-dependent significance test for parsimony. *Mol Phylogenet Evol* **4,** 64–71.
57. Yang, Z., and Roberts, D. (1995) On the use of nucleic acid sequences to infer early branches in the tree of life. *Mol Biol Evol* **12,** 451–58.
58. Gu, X., and Li, W.-H. (1996) Bias-corrected paralinear and logdet distances and tests of molecular clocks and phylogenies under nonstationary nucleotide frequencies. *Mol Biol Evol* **13,** 1375–83.
59. Gu, X., and Li, W.-H. (1998) Estimation of evolutionary distances under stationary and nonstationary models of nucleotide substitution. *Proc Natl Acad Sci USA* **95,** 5899–905.
60. Galtier, N., and Gouy, M. (1998) Inferring pattern and process: maximum-likelihood implementation of a nonhomogenous model of DNA sequence evolution for phylogenetic analysis. *Mol Biol Evol* **15,** 871–79.
61. Galtier, N., Tourasse, N., and Gouy, M. (1999) A nonhyperthermophilic common ancestor to extant life forms. *Science* **283,** 220–21.
62. Tamura, K., and Kumar, S. (2002) Evolutionary distance estimation under heterogeneous substitution pattern among lineages. *Mol Biol Evol* **19,** 1727–36.
63. Foster, P. G. (2004) Modelling compositional heterogeneity. *Syst Biol* **53,** 485–95.
64. Thollesson, M. (2004) LDDist: a Perl module for calculating LogDet pair-wise distances for protein and nucleotide sequences. *Bioinformatics* **20,** 416–18.
65. Lanave, C., Tommasi, S., Preparata, G., and Saccone, C. (1986) Transition and transversion rate in the evolution of animal mitochondrial DNA. *Bio Systems* **19,** 273–83.
66. Hashimoto, T., Nakamura, Y., Nakamura, F., Shirakura, T., Adachi, J., Goto, N., Okamoto, K.-I., and Hasegawa, M. (1994) Protein phylogeny gives a robust estimation for early divergences of eukaryoptes: phylogenetic place of a mitochondria-lacking protozoan, *Giardia lamblia*. *Mol Biol Evol* **11,** 65–71.
67. Hashimoto, T., Nakamura, Y., Kamaishi, T., Nakamura, F., Adachi, J., Okamoto, K.-I., and Hasegawa, M. (1995) Phylogenetic place of mitochondrial-lacking protozoan, *Giardia lamblia*, inferred from amino acid sequences of elongation factor 2. *Mol Biol Evol* **12,** 782–93.
68. von Haeseler, A., Janke, A., and Pääbo, S. (1993) Molecular phylogenetics. *Verhandlung der Deutschen Zoologischen Gesellschaft* **86,** 119–29.

69. Preparata, G., and Saccone, C. (1987) A simple quantitative model of the molecular clock. *J Mol Evol* **26,** 7–15.
70. Dowton, M., and Austin, A. D. (1997) The evolution of strand-specific compositional bias. A case study in the hymenopteran mitochondrial 16S rRNA gene. *Mol Biol Evol* **14,** 109–12.
71. Swofford, D. L. (2002), PAUP*: Phylogenetic analysis using parsimony (* and other methods), Version 4 Sinauer Associates, Sunderland, Massachusetts.
72. Strimmer, K., and von Haeseler, A. (1996) Quartet puzzling: a quartet maximum likelihood method for reconstructing tree topologies. *Mol Biol Evol* **13,** 964–69.
73. Schmidt, H. A., Strimmer, K., Vingron, M., and von Haeseler, A. (2002) TREE-PUZZLE: maximum likelihood phylogenetic analysis using quartets and parallel computing. *Bioinformatics* **18,** 502–04.
74. Bowker, A. H. (1948) A test for symmetry in contingency tables. *J Am Stat Assoc* **43,** 572–74.
75. Stuart, A. (1955) A test for homogeneity of the marginal distributions in a two-way classification. *Biometrika* **42,** 412–16.
76. Tavaré, S. (1986) Some probabilistic and statistical problems on the analysis of DNA sequences. *Lect Math Life Sci* **17,** 57–86.
77. Lanave, C., and Pesole, G. (1993) Stationary MARKOV processes in the evolution of biological macromolecules. *Binary* **5,** 191–95.
78. Waddell, P. J., and Steel, M. A. (1997) General time reversible distances with unequal rates across sites: mixing Γ and inverse Gaussian distributions with invariant sites. *Mol Phylogenet Evol* **8,** 398–414.
79. Waddell, P. J., Cao, Y., Hauf, J., and Hasegawa, M. (1999) Using novel phylogenetic methods to evaluate mammalian mtDNA, including amino acid-invariant sites-LogDet plus site stripping, to detect internal conflicts in the data, with special reference to the positions of hedgehog, armadillo, and elephant. *Syst Biol* **48,** 31–53.
80. Zhang, Y., Chen, M., Zhou, B. B., Jermiin, L. S., and Larkum, A. W. D. (2007) Evolution of the inner light-harvesting antennae protein family of cyanobacteria, algae and plants. *J Mol Evol* **64,** 321–31.
81. Beiko, R. G., and Charlebois, R. L. (2007) A simulation test bed for hypotheses of genome evolution. *Bioinformatics* **23,** 825–31.
82. Rodríguez-Ezpeleta, N., Brinkmann, H., Roure, B., Lartillot, N., Lang, B. F., and Philippe, H. (2007) Overcoming systematic errors in genome-scale phylogenies. *Syst Biol* **56,** 389–99.
83. Rzhetsky, A., and Nei, M. (1995) Tests of applicability of several substitution models for DNA sequence data. *Mol Biol Evol* **12,** 131–51.
84. Kumar, S., and Gadagkar, S. R. (2001) Disparity index: a simple statistic to measure and test the homogeneity of substitution patterns between molecular sequences. *Genetics* **158,** 1321–27.
85. Kumar, S., and Gadagkar, S. R. (2001) Corrigendum—disparity index: a simple statistic to measure and test the homogeneity of substitution patterns between molecular sequences. *Genetics* **159,** 913–14.
86. Phillips, M. J., and Penny, D. (2003) The root of the mammalian tree inferred from whole mitochondrial genomes. *Mol Phylogenet Evol* **28,** 171–85.
87. Susko, E., and Roger, A. J. (2007) On reduced amino acid alphabets for phylogenetic inference. *Mol Biol Evol* **24,** 2139–50.
88. Marquez, R., Smit, S., and Knight, R. (2005) Do universal codon-usage patterns minimize the effects of mutation and translation error? *Genome Biol* **6,** R91.
89. Smit, S., Yarus, M., and Knight, R. (2006) Natural selection is not required to explain universal compositional patterns in rRNA secondary structure categories. *RNA* **12,** 1–14.
90. Bourlat, S. J., Juliusdottir, T., Lowe, C. J., Freeman, R., Aronowicz, J., Kirschner, M., Lander, E. S., Thorndyke, M., Nakano, H., Kohn, A. B., Heyland, A., Moroz, L. L., Copley, R. R., and Telford, M. J. (2006) Deuterostome phylogeny reveals monophyletic chordates and the new phylum Xenoturbellida. *Nature* **444,** 85–88.
91. Woese, C. R., Achenbach, L., Rouviere, P., and Mandelco, L. (1991) Archaeal phylogeny: reexamination of the phylogenetic position of *Archaeoglobus fulgidus* in light of certain composition-induced artifacts. *Syst Appl Microbiol* **14,** 364–71.
92. Millen, R. S., Olmstead, R. G., Adams, K. L., Palmer, J. D., Lao, N. T., Heggie, L., Kavanagh, T. A., Hibberd, J. M., Gray, J. C., Morden, C. W., Calie, P. J., Jermiin, L. S., and Wolfe, K. H. (2001) Many parallel losses of *infA* from chloroplast DNA during angiosperm evolution with multiple independent transfers to the nucleus. *Plant Cell* **13,** 645–58.

93. Phillips, M. J., Lin, Y. H., Harrison, G. L., and Penny, D. (2001) Mitochondrial genomes of a bandicoot and a brushtail possum confirm the monophyly of australidelphian marsupials. *Proc R Soc Lond B Biol Sci* **268,** 1533–38.
94. Phillips, M., Delsuc, F., and Penny, D. (2004) Genome-scale phylogeny and the detection of systematic biases. *Mol Biol Evol* **21,** 1455–58.
95. Gibson, A., Gowri-Shankar, V., Higgs, P. G., and Rattray, M. (2005) A comprehensive analysis of mammalian mitochondrial genome base composition and improved phylogenetic methods. *Mol Biol Evol* **22,** 251–64.
96. Cannings, C., and Edwards, A. W. F. (1968) Natural selection and the de Finetti diagram. *Ann Hum Genet* **31,** 421–28.
97. Adkins, R. M., and Honeycutt, R. L. (1994) Evolution of the primate cytochrome c oxidase subunit II gene. *J Mol Evol* **38,** 215–31.
98. Amano, K., Nakamura, H., and Ichikawa, H. (2003) Self-organizing clustering: a novel non-hierarchical method for clustering large amount of DNA sequences. *Genome Informatics* **14,** 575–76.
99. Halanych, K. M., Bacheller, J. D., Aguinaldo, A. M. A., Liva, S. M., Hillis, D. M., and Lake, J. A. (1995) Evidence from 18S ribosomal DNA that the lophophorates are protostome animals. *Science* **267,** 1641–43.
100. Aguinaldo, A. M. A., Turbeville, J. M., Linford, L. S., Rivera, M. C., Garey, J. R., Raff, R. A., and Lake, J. A. (1997) Evidence for a clade of nematodes, arthropods and other moulting animals. *Nature* **387,** 489–93.
101. Giribet, G. (2002) Current advances in the phylogenetic reconstruction of metazoan evolution. A new paradigm for the Cambrian explosion? *Mol Phylogenet Evol* **24,** 345–57.
102. Halanych, K. M. (2004) The new view of animal phylogeny. *Annu Rev Ecol Evol Systemat* **35,** 229–56.
103. Philippe, H., Snell, E. A., Bapteste, E., Lopez, P., Holland, P. W. H., and Casane, D. (2004) Phylogenomics of eukaryotes: impact of missing data on large alignments. *Mol Biol Evol* **21,** 1740–52.
104. Philip, G. K., Creevey, C. J., and McInerney, J. O. (2005) The Opisthokonta and the Ecdysozoa may not be clades: Stronger support for the grouping of plant and animal than for animal and fungi and stronger support for the Coelomata than Exdysozoa. *Mol Biol Evol* **22,** 1175–84.
105. Rokas, A., Krüger, D., and Carroll, S. B. (2005) Animal evolution and the molecular signature of radiations compressed in time. *Science* **310,** 1933–38.
106. Philippe, H., Lartillot, N., and Brinkmann, H. (2005) Multigene analysis of bilaterian animals corroborate the monophyly of Ecdysozoa, Lophotrochozoa and Protostomia. *Mol Biol Evol* **22,** 1246–53.
107. Delsuc, F., Brinkmann, H., Chourrout, D., and Philippe, H. (2006) Tunicates and not cephalochordates are the closest living relatives of vertebrates. *Nature* **439,** 965–68.
108. Philippe, H., Brinkmann, H., Martinez, P., Riutort, M., and Baguñà, J. (2007) Acoel flatworms are not platyhelminthes: evidence from phylogenomics. *PLoS One* **2,** e717.
109. Baurain, D., Brinkmann, H., and Philippe, H. (2007) Lack of resolution in the animal phylogeny: closely spaced cladogenesis or undetected systematic errors? *Mol Biol Evol* **24,** 6–9.
110. Dunn, C. W., Hejnol, A., Matus, D. Q., Pang, K., Browne, W. E., Smith, S. A., Seaver, E., Rouse, G. W., Obst, M., Edgecombe, G. D., Sørensen, M. V., Haddock, S. H. D., Schmidt-Rhaesa, A., Okusu, A., Kristensen, R. M., Wheeler, W. C., Martindale, M. Q., and Giribet, G. (2008) Broad phylogenomic sampling improves resolution of the animal tree of life. *Nature* **452,** 745–50.

Chapter 5

Selection of Models of DNA Evolution with jModelTest

David Posada

Abstract

jModelTest is a bioinformatic tool for choosing among different models of nucleotide substitution. The program implements five different model selection strategies, including hierarchical and dynamical likelihood ratio tests (hLRT and dLRT), Akaike and Bayesian information criteria (AIC and BIC), and a performance-based decision theory method (DT). The output includes estimates of model selection uncertainty, parameter importances, and model-averaged parameter estimates, including model-averaged phylogenies. jModelTest is a Java program that runs under Mac OSX, Windows, and Unix systems with a Java Run Environment installed, and it can be freely downloaded from http://darwin.uvigo.es.

Key words: Model selection, likelihood ratio tests, AIC, BIC, performance-based selection, statistical phylogenetics.

1. Introduction

Phylogenetic reconstruction from DNA sequences is a problem of statistical inference. Since statistical inferences cannot be drawn in the absence of probabilities, the use of models of nucleotide substitution to calculate probabilities of change between nucleotides along the branches of a phylogenetic tree is essential for many evolutionary and comparative analyses. Importantly, the use of different models of DNA evolution can change the outcome of the phylogenetic analysis. The parameters affected include branch lengths, transition/transversion ratio, overall divergence, and rate variation among sites, whose estimates can be biased under a wrong model

David Posada (ed.), *Bioinformatics for DNA Sequence Analysis,* Methods in Molecular Biology 537

DOI 10.1007/978-1-59745-251-9_5

(1, 2). In addition, measures of confidence, hypothesis testing, and the phylogeny itself tend to be less accurate when the model assumed is grossly incorrect *(3–8)*.

Therefore, to have confidence in the inferences it is necessary to have confidence in the models. Several statistical strategies exist to justify the use of a particular model *(9, 10)*, although they tend to result in similar phylogenetic inferences at least under the Bayesian and maximum likelihood frameworks *(11, 12)*. Below we describe the model selection strategies implemented in the program jModelTest (*see* **Note 1**).

1.1. Likelihood and Model Fit

Model selection strategies use the likelihood function as a measure of model fit. The likelihood is proportional to the probability of the data (D), given a model of evolution (M), a vector of K model parameters (θ), a tree topology (τ), and a vector of branch lengths (ν):

$$L = P(D|M, \theta, \tau, \nu)$$

Because in this case we are really interested in the model but not in the tree or in the parameter values, we use the maximum likelihood estimates (*see* **Note 2**). In addition, and for computational reasons, we usually work with the maximized log likelihood:

$$\ell = \ln P(D\Big|M, \hat{\theta}, \hat{\tau}, \hat{\nu})$$

1.2. Sequential Likelihood Ratio Tests

In traditional statistical theory, a widely accepted statistic for testing the relative fit of two models is the likelihood ratio test statistic (LRT):

$$LRT = 2(\ell_1 - \ell_0)$$

where ℓ_1 is the maximum likelihood under the more parameter-rich, complex model and ℓ_0 is the maximum likelihood under the less parameter-rich, simple model. When the models compared are nested – the simple model a special case of the complex model – the LRT statistic is asymptotically distributed as a χ^2 with a number of degrees of freedom equal to the difference in the number of free parameters between the two models *(13, 14)* (*see* **Note 3**).

Importantly, a sequence of LRTs can be performed by adding (*forward selection*) or removing parameters (*backward selection*) in a particular hierarchy. This strategy has been named hierarchical likelihood ratio tests or ***hLRTs*** (**Fig. 5.1**). Alternatively, the order in which parameters are added or removed can be selected automatically – dynamical likelihood ratio tests or ***dLRTs*** (**Fig. 5.2**) – for example, by adding the parameter that

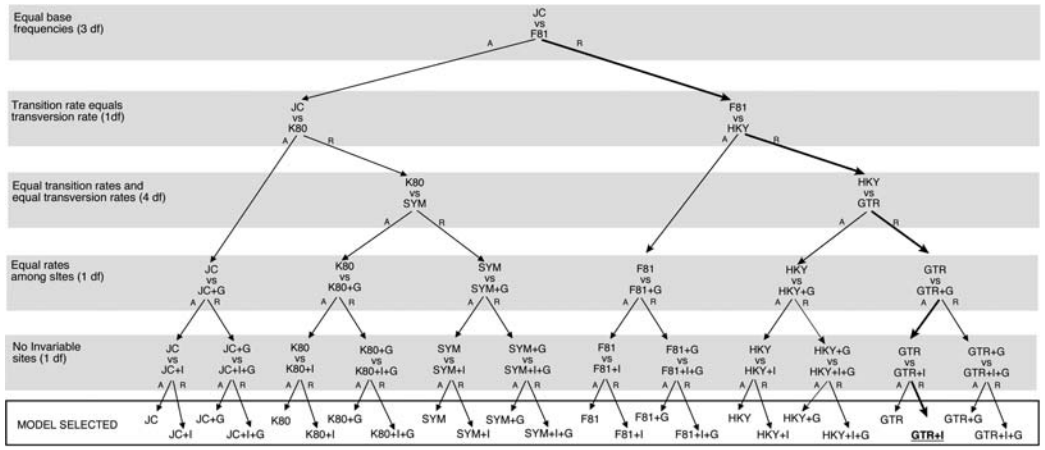


Fig. 5.1. Example of a particular forward hierarchy of likelihood ratio tests for 24 models. At any level the null hypothesis (model on top) is either accepted (A) or rejected (R). The model selected here is GTR+I.

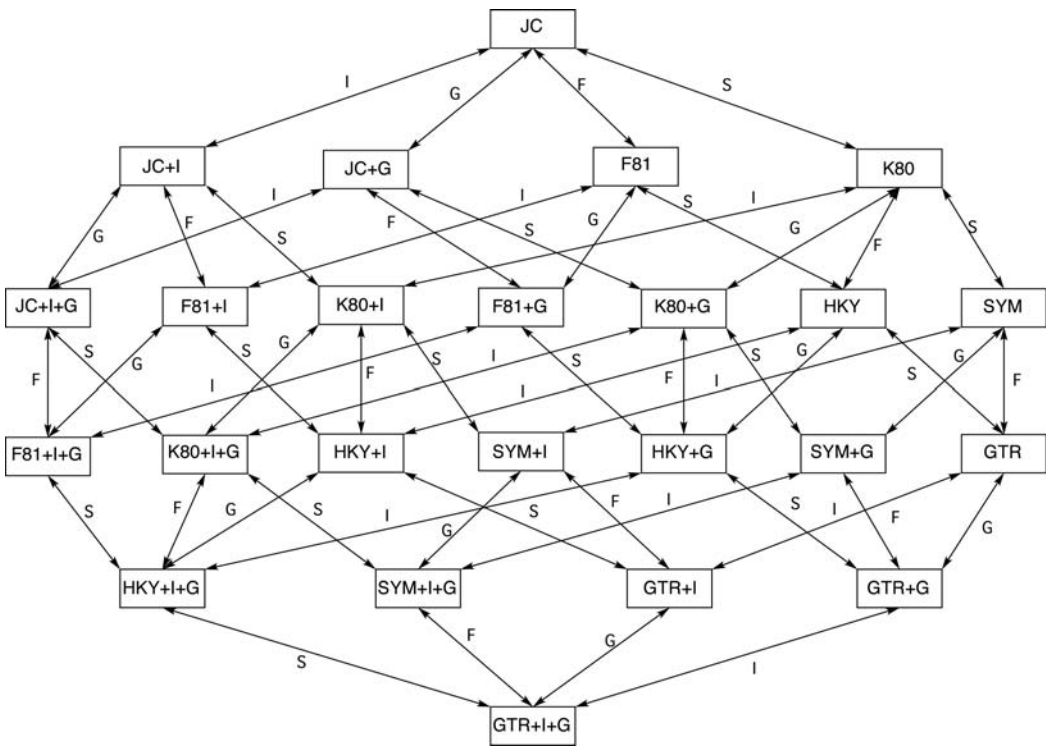


Fig. 5.2. Dynamical likelihood ratio tests. Starting from the simplest (*top; forward selection*) or most complex model (*bottom; backward selection*), the path which results in the highest significant change of the likelihood during forward selection – or the smallest non-significant change of the likelihood during forward selection – is followed until no further movements are possible. I: proportion of invariable sites; G: rate variation among sites; F: base frequencies; S: substitution parameters.

maximizes a significant gain in likelihood during forward selection, or to add the parameter that minimizes a non-significant loss in likelihood during backward selection *(15)*. In this case, the order of the tests is not specified a priori, but it will depend on the particular data.

Although sequential LRTs are possibly one of the most popular strategies for the selection of models of DNA evolution, Posada and Buckley *(9)* have pointed out some important problems associated to the use of pairwise LRTs for model selection in this context.

1.3. Akaike Information Criterion

A different approach for model selection is the simultaneous comparison of all competing models. This can be accomplished with the Akaike information criterion (AIC), *(16)*, which is an asymptotically unbiased estimator of the Kullback-Leibler information quantity *(17)* (*see* **Note 4**). We can think of the AIC as the amount of information lost when using models to approximate the process of molecular evolution, so the model with the smallest AIC is preferred:

$$AIC = -2\ell + 2\,K,$$

where K is the number of free parameters in the model, including branch lengths. The AIC can be used to compare both nested and non-nested models. When sample size (n) (*see* **Note 5**) is small compared to the number of parameters (say, $n/K < 40$), the AICc *(18, 19)* is recommended instead:

$$AIC_c = AIC + \frac{2\,K(K+1)}{n-K-1}$$

1.4. Bayesian Information Criterion

A Bayesian alternative to the use of the AIC is the Bayesian Information Criterion (BIC) *(20)* (*see* **Note 6**):

$$BIC = -2\ell + K \log n$$

Choosing the model with the smallest BIC is equivalent to selecting the model with the maximum posterior probability, given equal priors for all competing models. If we take the sample size n to be the length of the alignment, for typical alignments the natural log of n will be usually > 2, and the BIC will tend to choose simpler models than the AIC. Indeed, this has been observed in practice. Like the AIC, the BIC can be used to compare nested and non-nested models. Also, BIC differences or BIC weights can easily be calculated.

1.5. Performance-Based Selection

Minin et al. *(21)* developed a decision theoretic approach (DT) that selects models on the basis of their phylogenetic performance, measured as the expected error on branch lengths estimates weighted by their BIC. The best DT model is the one that minimizes the risk function:

$$C_i \approx \sum_{j=1}^{R} \left\| \hat{\mathbf{B}}_i - \hat{\mathbf{B}}_j \right\| \frac{e^{-BIC_i/2}}{\sum_{j=1}^{R} e^{-BIC_i/2}},$$

where

$$\left\| \hat{\mathbf{B}}_i - \hat{\mathbf{B}}_j \right\|^2 = \sum_{l=1}^{2t-3} (\hat{B}_{il} - \hat{B}_{jl})^2$$

and where t is the number of taxa. Simulations suggest that models selected with this criterion result in slightly more accurate branch length estimates than those obtained under models selected by the hLRTs *(21, 22)*.

1.6. Model Uncertainty

By calculating the differences (Δ) between any model and the best model under the AIC, BIC, or DT strategies, it is possible to get a sense of the confidence in the model selected. For example, for the ith model, the AIC difference is

$$\Delta_i = AIC_i - min(AIC)$$

These differences are easy to interpret and allow for a quick comparison and ranking of candidate models (*see* **Note 7**). More importantly these differences can be used to obtain the relative weight of each model:

$$w_i = \frac{\exp(-1/2\Delta_i)}{\sum_{r=1}^{R} \exp(-1/2\Delta_r)},$$

which can be interpreted, from a Bayesian perspective, as the probability that a model is the best approximation to the truth given the data (*see* **Note 8**). The weights for every model add to 1, so we can establish an approximate confidence set of models for the best models by summing the weights from largest to smallest until the sum is above some threshold *(23,* pp. 169–171, *24)*. This interval can also be set up stochastically; when the confidence interval (CI) includes only partially a given model, this model is included in the CI with a probability equal to the fraction included.

1.7. Model Averaging

Often there is some uncertainty in selecting the best candidate model. In such cases, or just when one does not want to rely on a single model, inferences can be drawn from all models (or an optimal subset) simultaneously. This is known as *model*

averaging or *multimodel inference*. Within the AIC or Bayesian frameworks, it is straightforward to obtain a model-averaged estimate of any parameter *(24–29)*. For example, a model-averaged estimate of the substitution rate between adenine and cytosine (φ_{A-C}) using the Akaike weights (w_i) for R candidate models would be

$$\hat{\bar{\varphi}}_{A-C} = \frac{\sum_{i=1}^{R} w_i I_{\varphi_{A-C}}(M_i)\varphi_{A-C_i}}{w_+(\varphi_{A-C})},$$

where

$$w_+(\varphi_{A-C}) = \sum_{i=1}^{R} w_i I_{\varphi_{A-C}}(M_i),$$

and

$$I_{\varphi_{A-C}}(M_i) = \begin{cases} 1 & \text{if } \varphi_{A-C} \text{ is in model } M_i \\ 0 & \text{otherwise} \end{cases}$$

Importantly, the averaged parameter could be the topology itself, so it is possible to construct a model-averaged estimate of the phylogeny by estimating a maximum likelihood tree for every model and building a weighted consensus tree of these using the corresponding AIC or BIC weights, for example.

1.8. Parameter Importance

The *relative importance* of any parameter can be estimated by summing the weights across all models that include such a parameter. For example, the relative importance of the substitution rate between adenine and cytosine across all candidate models is simply the denominator above, $w_+(\varphi_{A-C})$ (*see* **Note 9**).

2. Program Usage

jModelTest is a Java program developed in Xcode under MacOS X 10.5. The program displays a GUI where the user can select the input file (a DNA alignment) and specify the different options for the model selection analysis. To accomplish most of its tasks, jModelTest builds up a pipeline with several freely available programs:

- ReadSeq *(30)*: to read the DNA alignment.
- Phyml *(31)*: for the likelihood calculations.
- Consense *(32)*: to build a consensus tree representing the model-averaged phylogeny.
- Ted (D. Posada): to calculate pairwise Euclidean distances between trees for the computation of the DT score.

2.1. Starting the Program

The *jModelTest.jar* file should be executable in any OS with a Java Runtime Environment (*see* **Note 10**). For the accompanying programs, executables are provided for MacOS X, Windows XP, and Linux. After double-clicking on the jar file, the program console, with several menus and a text panel, should open. If this does not work, the program can be started from the command prompt, after moving to the jModelTest folder and typing "java –jar jModeltest.1.0.jar"). The text in the program console can be edited ("Edit > ...") and saved to a file ("Edit > Save console"), or printed ("Edit > Print console") at any time.

2.2. Input Data Files

The input file for jModelTest is a DNA sequence alignment in any of the formats accepted by ReadSeq *(30)* (*see* **Note 11**), including FASTA, PHYLIP, and NEXUS. The input file can be specified by clicking on the menu "File > Load DNA alignment."

2.3. Likelihood Settings

There are 88 models currently implemented in jModelTest, including 11 substitution schemes, equal or unequal base frequencies (+F), a proportion of invariable sites (+I), and rate variation among sites with a number of rate categories (+G) (**Table 5.1**). The panel for the likelihood calculations is available from the menu Analysis > "Compute likelihood scores" (**Fig. 5.3**). It is possible to specify which models will be compared to some extent, from 3 to 88 models.

Table 5.1
Substitution models available in jModelTest. Any of these models can include invariable sites (+I), rate variation among sites (+G), or both (+I+G)

Model	*Reference*	Free parameters	Base frequencies	Substitution rates	Substitution code
JC	*(39)*	0	Equal	AC=AG=AT=CG=CT=GT	000000
F81	*(40)*	3	Unequal	AC=AG=AT=CG=CT=GT	000000
K80	*(41)*	1	Equal	AC=AT=CG=GT; AG=CT	010010
HKY	*(42)*	4	Unequal	AC=AT=CG=GT; AG=CT	010010
TNef	*(43)*	2	Equal	AC=AT=CG=GT; AG; CT	010020
TN	*(43)*	5	Unequal	AC=AT=CG=GT; AG; CT	010020
TPM1= K81	*(44)*	2	Equal	AC=GT; AT=CG; AG=CT	012210
TPM1uf	*(44)*	5	Unequal	AC=GT; AT=CG; AG=CT	012210
TPM2		2	Equal	AC=AT; CG=GT; AG=CT	010212

(continued)

Table 5.1(continued)

Model	*Reference*	Free parameters	Base frequencies	Substitution rates	Substitution code
TPM2uf		5	Unequal	AC=AT; CG=GT; AG=CT	010212
TPM3		2	Equal	AC=CG; AT=GT; AG=CT	012012
TPM3uf		5	Unequal	AC=CG; AT=GT; AG=CT	012012
TIM1ef	*(25)*	3	Equal	AC=GT; AT=CG; AG, CT	012230
TIM1	*(25)*	6	Unequal	AC=GT; AT=CG; AG, CT	012230
TIM2ef		3	Equal	AC=AT; CG=GT; AG; CT	010232
TIM2		6	Unequal	AC=AT; CG=GT; AG; CT	010232
TIM3ef		3	Equal	AC=CG; AT=GT; AG; CT	012032
TIM3		6	Unequal	AC=CG; AT=GT; AG; CT	012032
TVMef	*(25)*	4	Equal	AC; AT, CG; GT; AG=CT	012314
TVM	*(25)*	7	Unequal	AC; AT; CG; GT; AG=CT	012314
SYM	*(45)*	5	Equal	AC; AG; AT; CG; CT; GT	012345
GTR = REV	*(46)*	8	Unequal	AC; AG; AT; CG; CT; GT	012345

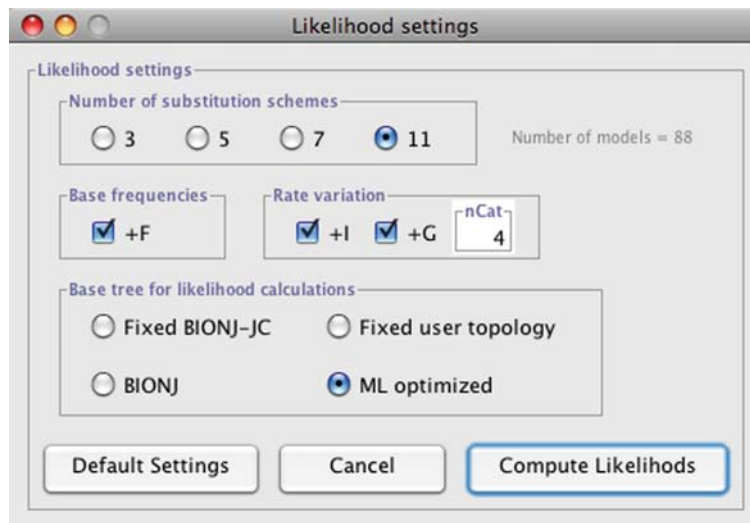


Fig. 5.3. Likelihood settings panel. The different options specify which parameters are considered in defining a candidate set of models and which base tree is used for the likelihood calculations.

For the likelihood calculations, there is the option of fixing the topology or to optimize it. In all cases branch lengths are estimated and counted as model parameters. A fixed tree can be estimated using the BIONJ algorithm *(33)* with the JC model, or it can be specified by the user from a file (in Newick format). Alternatively, potentially different BIONJ or ML trees can be estimated for each model, which will require more computation time, specially for the ML optimization.

2.4. Model Selection and Averaging

Once the likelihood calculations have finished, the options for the different model selection strategies can be specified from the menu ("Analysis > . . .").

2.4.1. Sequential LRTs

The user can build a particular hierarchy of LRTs by specifying the order of the tests and whether parameters are added (forward selection) or removed (backward selection) (hLRTs) (**Fig. 5.4**). Alternatively, the order of the LRTs can be set automatically (dLRTs). The LRTs will be available only if the likelihoods scores were previously calculated upon a fixed topology, due to the nesting requirement of the χ^2 approximation.

2.4.2. AIC, BIC, and DT

Under the AIC framework, the user can select whether to use the AICc or the AIC, which is the default (**Fig. 5.5**). If the AICc is specified, sample size will have to be indicated, although by default it is the number of sites in the alignment. A CI of models including a specified fraction of the models – by default 100% – will also be built according to the cumulative weight. A block of PAUP* *(34)* commands specifying the likelihood settings of the AIC model can

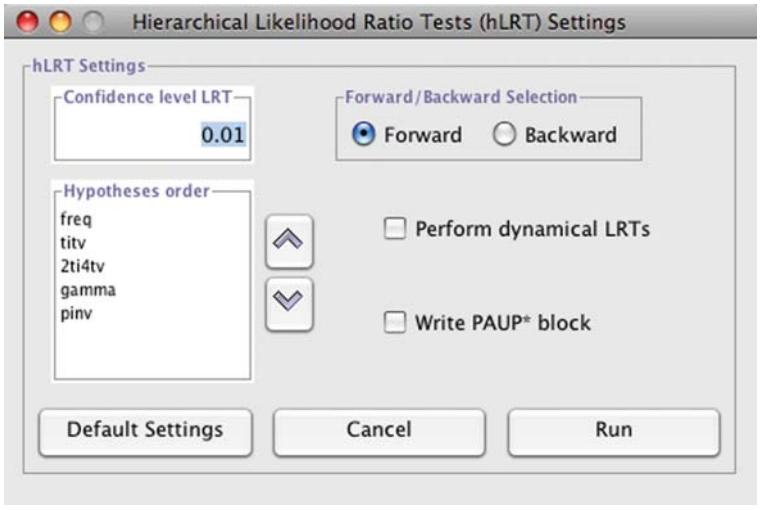


Fig. 5.4. Likelihood ratio tests panel. Here the user will establish the different options for the hLRT or dLRT analysis, including the order and direction of tests. The confidence level for each LRT is 0.1 by default.

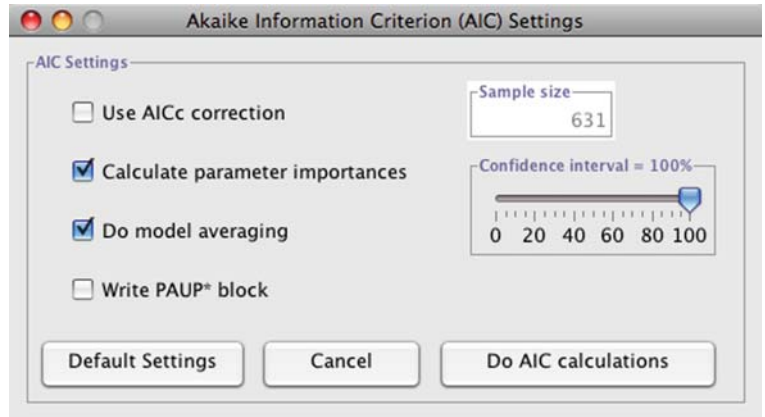


Fig. 5.5. AIC panel. This window is used to configure the AIC or AICc analysis, including the calculation of confidence interval, model averaging estimates, and parameter importances.

be written to the console. The options for the BIC and DT selection schemes are very similar, expect for the lack of a correction for small samples.

2.4.3. Model Averaged Phylogeny

A model averaged estimate of the tree topology can be obtained by calculating a weighted (using will be AIC, BIC or DT weights) consensus (*majority rule* or *strict*) (*see* **Note 12**) from all the trees corresponding to the models in the candidate set, or within a given CI (**Fig. 5.6**).

2.4.4. Other Options

The program includes a very simple calculator to perform LRTs using the standard or a mixed χ^2 approximation (**Fig. 5.7**). The models tested should be nested. In addition, the likelihood scores and the results from the different analyses are stored in a table that can be displayed at any time from the menu "Results > Show results table" (**Fig. 5.8**).

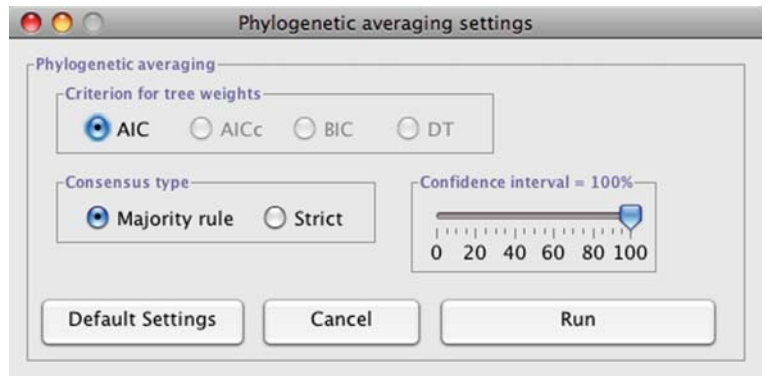


Fig. 5.6. Phylogenetic averaging panel. The user controls here how to obtain a model-averaged estimate of the phylogeny.

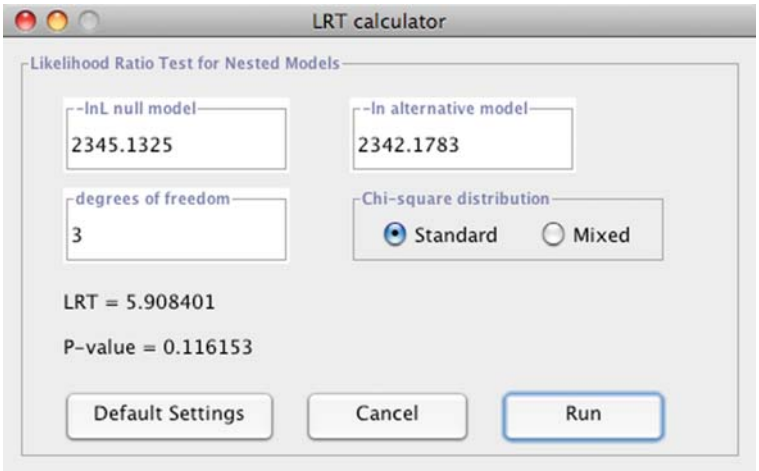
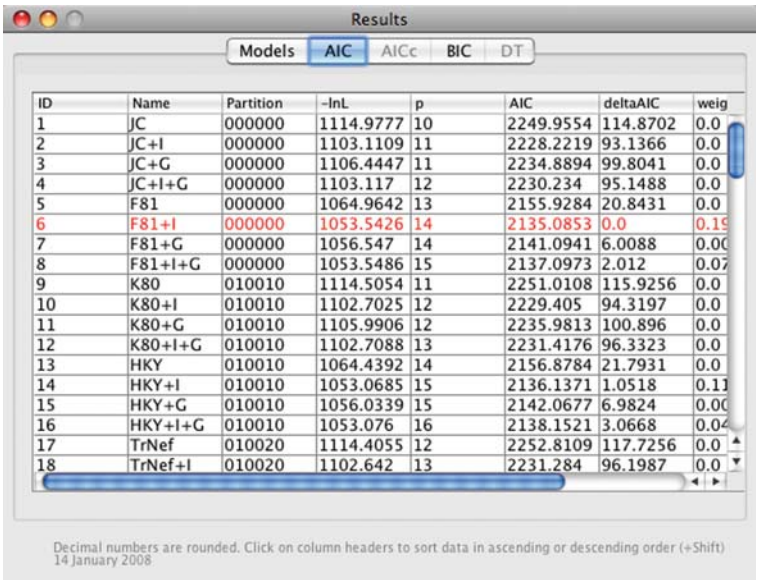


Fig. 5.7. LRT calculator. This simple tool allows for the fast calculation of pairwise LRTs.



ID	Name	Partition	-lnL	p	AIC	deltaAIC	weig
1	JC	000000	1114.9777	10	2249.9554	114.8702	0.0
2	JC+I	000000	1103.1109	11	2228.2219	93.1366	0.0
3	JC+G	000000	1106.4447	11	2234.8894	99.8041	0.0
4	JC+I+G	000000	1103.117	12	2230.234	95.1488	0.0
5	F81	000000	1064.9642	13	2155.9284	20.8431	0.0
6	F81+I	000000	1053.5426	14	2135.0853	0.0	0.1
7	F81+G	000000	1056.547	14	2141.0941	6.0088	0.0
8	F81+I+G	000000	1053.5486	15	2137.0973	2.012	0.0
9	K80	010010	1114.5054	11	2251.0108	115.9256	0.0
10	K80+I	010010	1102.7025	12	2229.405	94.3197	0.0
11	K80+G	010010	1105.9906	12	2235.9813	100.896	0.0
12	K80+I+G	010010	1102.7088	13	2231.4176	96.3323	0.0
13	HKY	010010	1064.4392	14	2156.8784	21.7931	0.0
14	HKY+I	010010	1053.0685	15	2136.1371	1.0518	0.1
15	HKY+G	010010	1056.0339	15	2142.0677	6.9824	0.0
16	HKY+I+G	010010	1053.076	16	2138.1521	3.0668	0.0
17	TrNef	010020	1114.4055	12	2252.8109	117.7256	0.0
18	TrNef+I	010020	1102.642	13	2231.284	96.1987	0.0

Fig. 5.8. Calculation table. This table displays the different parameter and scores calculated during the analysis. The model selected is highlighted in a different color.

3. Examples

The example data set consists of an alignment of 35 DNA sequences 395 nucleotides long, of the *vpu* HIV-1 gene, including representative virus of non-recombinant subtypes

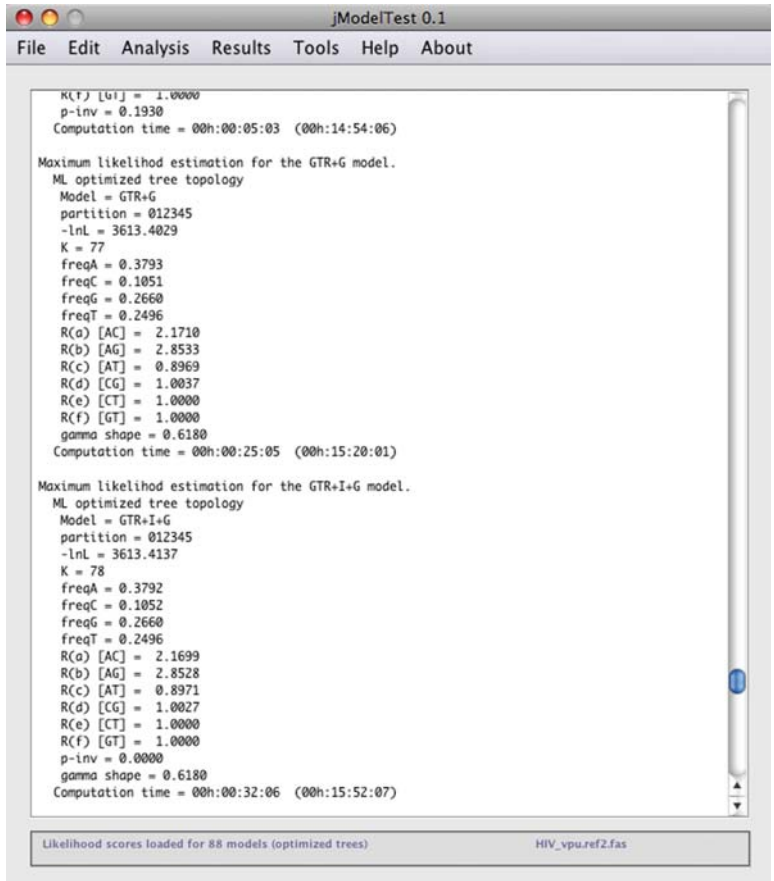


Fig. 5.9. Likelihood calculations for the HIV-1 data. The likelihood scores and parameter values for the last two models computed for the HIV-1 *vpu* alignment are displayed in the main jModelTest GUI.

within the HIV-1 group M (*see* **Note 13**). The alignment was downloaded in the FASTA format and loaded into jModelTest ("File > Load DNA alignment"). Likelihood scores were calculated for 88 models while optimizing the tree topology for each model by maximum likelihood ("Analysis > Compute likelihood scores") (**Fig. 5.9**). The computation took almost 16 minutes on the background on a 2.16 GHz Intel Core Duo MacBook Pro laptop running MacOS X 10.5.2.

Hierarchical likelihood ratio tests were carried out with the default hypothesis order and with forward addition of parameters, using a fixed BIONJ-JC tree topology for all calculations. The model finally selected was GTR+G (**Fig. 5.10**), which allows for unequal base frequencies, six substitution rates, and rate variation among sites (α = alpha shape of the gamma (G) distribution) (**Table 5.1**). Inspecting the results we notice that

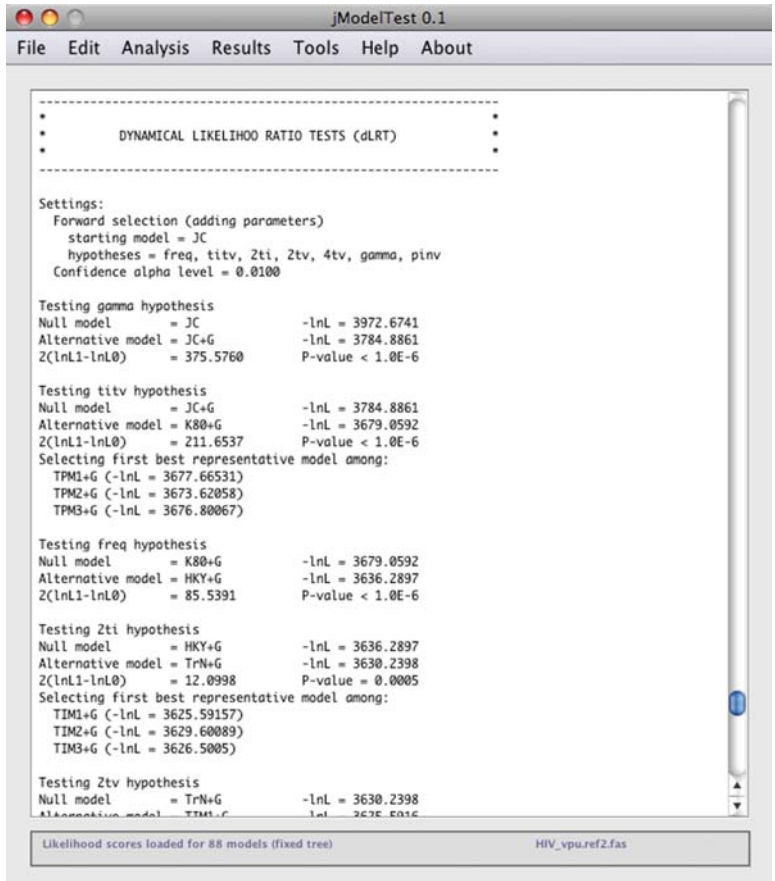


Fig. 5.10. Dynamical forward likelihood ratio tests for the HIV-1 alignment. All LRTs shown are strongly rejected.

the frequencies of A and C are quite distinct and certain substitutions are somehow more common than others, like A⇔C and A⇔G. The rate heterogeneity among sites is important [α(G) = 0.6270] ($\alpha < 1$ indicates strong rate variation). When backward selection was implemented, GTR+I+G was chosen instead, highlighting the relevance of the order of the LRTs on the final model selected. dLRTs with forward addition of parameters resulted in the selection of a third different model, TIM1+G, which is similar to the GTR+G except for the fact that it considers only two types of transversions (A⇔C = G⇔T; A⇔T = C⇔G). In this case, backward and forward selection resulted in the same model.

The AIC selected the GTR+G model as the best model among the 88 candidates, with a weight of 0.6361 (**Fig. 5.11**). The 95% CI included four models: GTR+G,

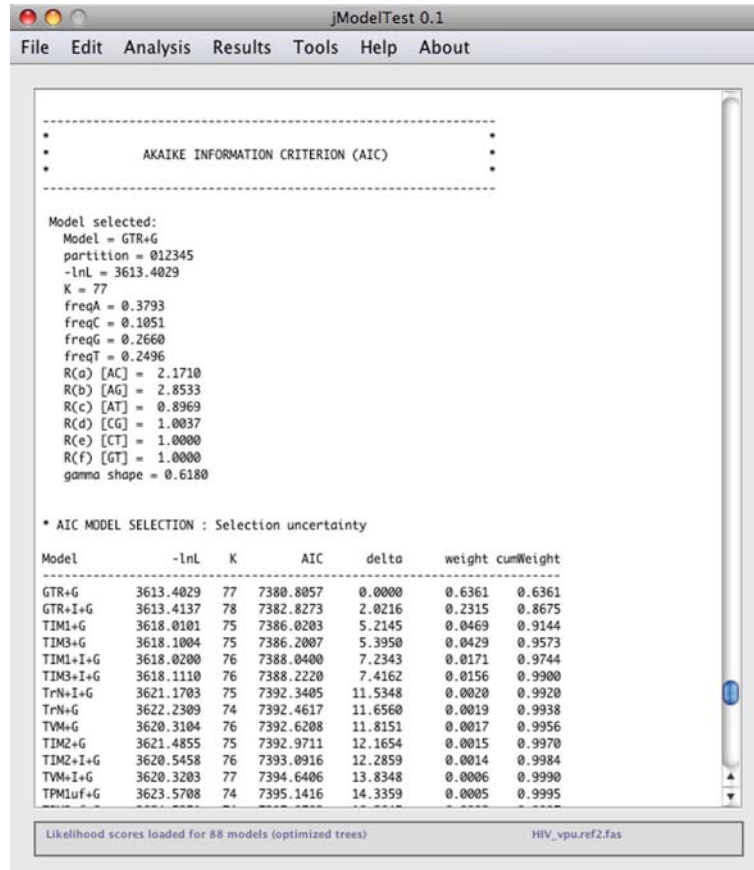


Fig. 5.11. AIC selection for the HIV-1 data. The model selected was GTR+G, with a AIC weight of 0.6361. –lnL: negative log likelihood; K: number of estimated parameters; AIC: Akaike Information Criterion; delta: AIC difference; weight: AIC weight; cumWeight: cumulative AIC weight.

GTR+I+G, TIM1+G, and TIM3+G. The parameter importances suggested that different base frequencies, substitution rates, and rate variation among sites need to be taken into account to describe the evolution of this gene (**Fig. 5.12**). The results for the AICc using the alignment length as sample size were almost identical. The AIC model averaged estimates resulted in similar values and trends to those already described for the GTR+G selected by the hLRTs. Approximately a fifth of the sites are invariable when rate heterogeneity is not considered (*p-inv* (I) = 0.1934), but none is invariable when rate heterogeneity is taken into account (*p-inv* (IG) = 0). This is an example of how the proportion of invariable site and the shape of the gamma-

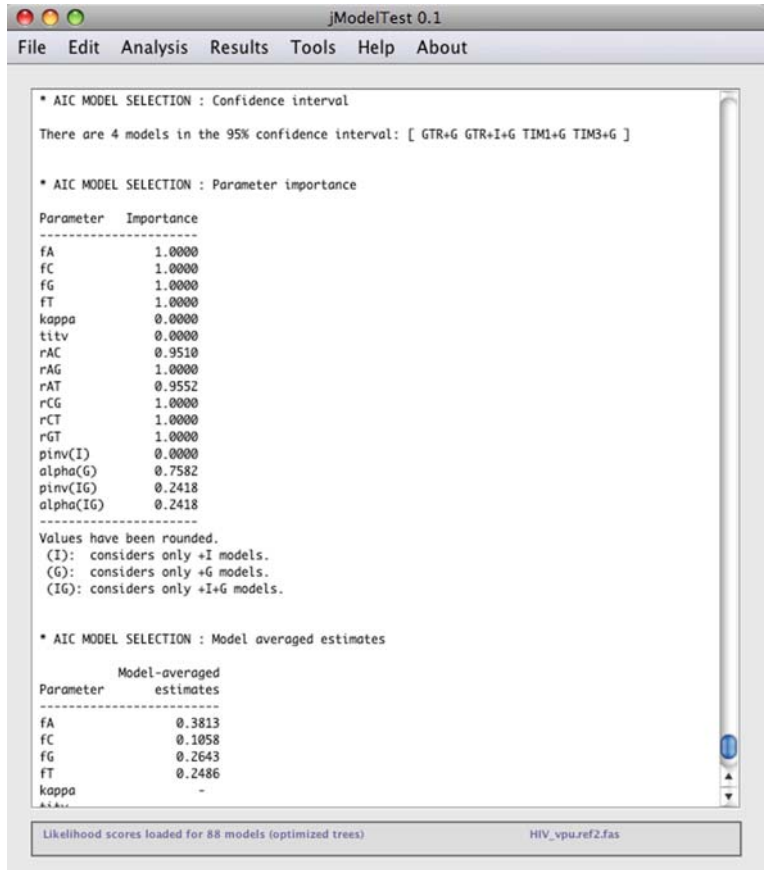


Fig. 5.12. AIC parameter importances and model averaged estimates for the HIV-1 dataset.

distributed rates interact together and therefore need to be interpreted in the light of each other. The rate heterogeneity is noticeable [α(G) = α(IG) = 0.6197].

The BIC selected a simpler model than the AIC (TIM1+G), but with a smaller weight (0.3583). The 95% CI encompasses eight models, but includes all the models listed within the AIC 95% CI. Parameter importances were also similar, although rate variation is deemed more important, as displayed almost exclusively by the +G models (α(G) = 0.9420). Model-averaged parameter estimates were very similar to those obtained under the AIC framework.

The DT selected the same model as the BIC (TIM1+G). The DT weights as calculated in jModelTest 0.1 are very experimental and not deemed very reliable, so model weights and parameter importances will not be not taken into

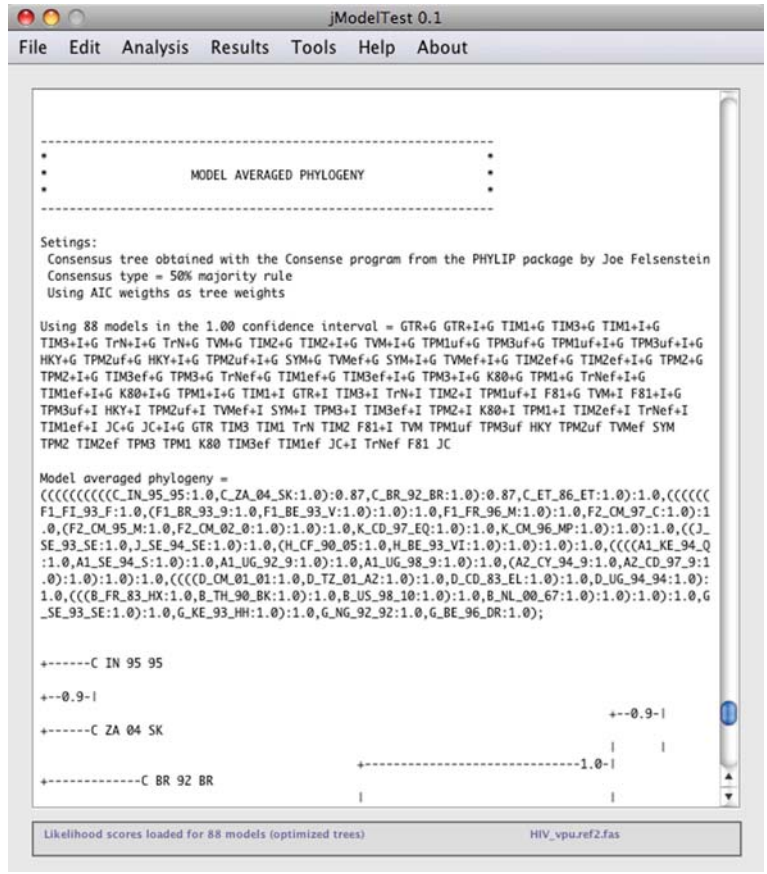


Fig. 5.13. AIC model-averaged phylogeny for the HIV-1 data. The program displays the output of the Consense program (Phylip package), including the Newick string representation of the tree and a simple tree plot. Values above branches represent the degree of phylogenetic uncertainty due to model selection (1.0 = no uncertainty).

account. In any case, model-averaged parameter estimates were very similar to those obtained by the AIC or BIC weights.

The model-averaged phylogenies were very similar regardless of the weights used and identical respecting the consensus type. In fact only two closely related averaged tree topologies were obtained (Robinson-Fould symmetrical distance = 4), one with the AIC and AICc weights and the other with the BIC and DT weights. Most clades received a weight support of 1.0, indicating that they are stable to phylogenetic uncertainty due to model selection (**Fig. 5.13**). The Newick strings provided by the program can be directly displayed in available tree-plotting programs (**Fig. 5.14**) (*see* **Note 14**).

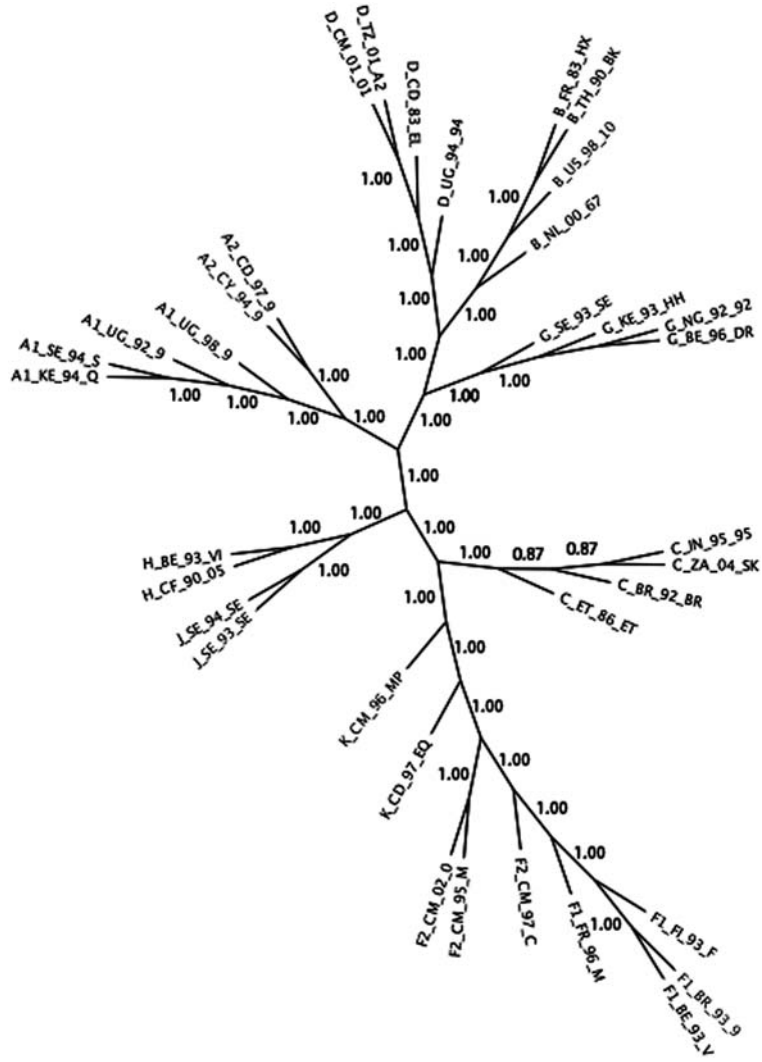


Fig. 5.14. AIC model averaged HIV-1 phylogeny represented in the program FigTree. Values above branches represent degree of uncertainty due to model selection (1.0 = no uncertainty).

4. Notes

1. In addition, although not considered here, the overall adequacy of a particular model can also be evaluated using different procedures *(13, 35)*.
2. Those parameters that we need to calculate, but that are not really what we want to infer, are called *nuisance* parameters. We can get rid of them by maximizing them – typically

under a maximum likelihood framework – or by integrating over their possible values – typically under a Bayesian setting.

3. To preserve the nesting of the models, the likelihoods need to be estimated upon the same tree topology. In addition, when some parameter is fixed at its boundary, a mixed χ^2 is used instead *(36, 37)*.
4. Burnham and Anderson *(24)* provide an excellent introduction to the AIC and model selection in general.
5. The effective sample size of an alignment of DNA sequences is poorly understood. Most researchers, when needed, use the length of the alignment as a surrogate for sample size. However, sites in an alignment are not independent, which should result in an overestimate. Using the number of variable sites or some measure of column variability (e.g., entropy) are alternative options that might underestimate sample size. In addition, it seems that the number of sequences should be taken into account somehow, although multiplying the number of sequences by the length should be an overestimate. Clearly, this is a topic that deserves further research.
6. Alternatively, Bayes factors for models of molecular evolution can be calculated using reversible jump MCMC *(38)*. Bayes factors are the Bayesian analogues of the LRTs.
7. As a rough rule of thumb, models having Δ_i within 1–2 of the best model have substantial support and should receive consideration. Models having Δ_i within 3–7 of the best model have considerably less support, while models with $\Delta_i > 10$ have essentially no support.
8. This equation will not work for the DT. This is because DT statistic is of a different nature, and the standard theory does not apply anymore. Right now in jModelTest, the DT weights are simply the rescaled reciprocal DT scores ((1/DTi)/sum). This is very gross and should be used with caution. Note that parameter importances and model averaged estimates also use these weights.
9. One needs to be careful when interpreting the relative importance of parameters. When the number of candidate models is less than the number of possible combinations of parameters, the presence–absence of some pairs of parameters can be correlated, and so their relative importances.
10. http://www.java.com/getjava/
11. http://iubio.bio.indiana.edu/soft/molbio/readseq/java/. Note that ReadSeq does not read in long sequence names in several formats.

12. For the strict consensus, tree weights have no meaning, as any clade in the strict consensus has to be in all trees in the collection.
13. This *vpu* subtype reference alignment was downloaded from the HIV database at Los Alamos National Laboratory (http://www.hiv.lanl.gov/content/sequence/HIV/SUBTYPE_REF /align.html). Putative recombinants were not downloaded, and two redundant sequences were removed from the alignment. This gene codifies a small accessory protein, which is inserted into the membranes of the infected cells, and which seems to be involved in the degradation of the CD4 receptor and in the release of viral particles.
14. See, for example, FigTree at http://tree.bio.ed.ac.uk/software/figtree/ or TreeView at http://taxonomy.zoology.gla.ac.uk/rod/treeview.html.

References

1. Yang, Z., Goldman, N., and Friday, A. (1995) Maximum likelihood trees from DNA sequences: a peculiar statistical estimation problem. *Syst Biol* **44**, 384–99.
2. Tamura, K. (1994) Model selection in the estimation of the number of nucleotide substitutions. *Mol Biol Evol* **11**, 154–57.
3. Zhang, J. (1999) Performance of likelihood ratio tests of evolutionary hypotheses under inadequate substitution models. *Mol Biol Evol* **16**, 868–75.
4. Lemmon, A. R., and Moriarty, E. C. (2004) The importance of proper model assumption in Bayesian phylogenetics. *Syst Biol* **53**, 265–77.
5. Buckley, T. R., and Cunningham, C. W. (2002) The effects of nucleotide substitution model assumptions on estimates of nonparametric bootstrap support. *Mol Biol Evol* **19**, 394–405.
6. Sullivan, J., and Swofford, D. L. (1997) Are guinea pigs rodents? The importance of adequate models in molecular phylogenies. *J Mamm Evol* **4**, 77–86.
7. Kelsey, C. R., Crandall, K. A., and Voevodin, A. F. (1999) Different models, different trees: the geographic origin of PTLV-I. *Mol Phylogenet Evol* **13**, 336–47.
8. Pupko, T., Huchon, D., Cao, Y., Okada, N., and Hasegawa, M. (2002) Combining multiple data sets in a likelihood analysis: which models are the best? *Mol Biol Evol* **19**, 2294–307.
9. Posada, D., and Buckley, T. R. (2004) Model selection and model averaging in phylogenetics: advantages of Akaike Information Criterion and Bayesian approaches over likelihood ratio tests. *Syst Biol* **53,** 793–808.
10. Sullivan, J., and Joyce, P. (2005) Model selection in phylogenetics. *Annu Rev Ecol Evol. Syst.* **36,** 445–66.
11. Alfaro, M. E., and Huelsenbeck, J. P. (2006) Comparative performance of Bayesian and AIC-based measures of phylogenetic model uncertainty. *Syst Biol* **55,** 89–96.
12. Ripplinger, J., and Sullivan, J. (2008) Does choice in model selection affect maximum likelihood analysis? *Syst Biol* **57,** 76–85.
13. Goldman, N. (1993) Statistical tests of models of DNA substitution. *J Mol. Evol* **36,** 182–98.
14. Kendall, M., and Stuart, A. (1979) The Advanced Theory of Statistics, Charles Griffin, London.
15. Posada, D., and Crandall, K. A. (2001) Selecting the best-fit model of nucleotide substitution. *Syst Biol* **50,** 580–601.
16. Akaike, H. (1974) A new look at the statistical model identification. *IEEE Trans. Aut. Control* **19,** 716–23.
17. Kullback, S., and Leibler, R. A. (1951) On information and sufficiency. *Ann Math Stat* **22,** 79–86.
18. Sugiura, N. (1978) Further analysis of the data by Akaike's information criterion and

the finite corrections. *Comm Statist Theor Meth* **A7,** 13–26.

19. Hurvich, C. M., and Tsai, C.-L. (1989) Regression and time series model selection in small samples. *Biometrika* **76,** 297–307.
20. Schwarz, G. (1978) Estimating the dimension of a model. *Ann Stat* **6,** 461–64.
21. Minin, V., Abdo, Z., Joyce, P., and Sullivan, J. (2003) Performance-based selection of likelihood models for phylogeny estimation. *Syst Biol* **52,** 674–83.
22. Abdo, Z., Minin, V. N., Joyce, P., and Sullivan, J. (2005) Accounting for uncertainty in the tree topology has little effect on the decision-theoretic approach to model selection in phylogeny estimation. *Mol Biol Evol* **22,** 691–703.
23. Burnham, K. P., and Anderson, D. R. (1998) *Model Selection and Inference: A Practical Information-Theoretic Approach*, Springer-Verlag, New York, NY.
24. Burnham, K. P., and Anderson, D. R. (2003) *Model Selection and Multimodel Inference: A Practical Information-Theoretic Approach*, Springer-Verlag, New York, NY.
25. Posada, D. (2003) *Current Protocols in Bioinformatics* (Baxevanis, A. D., Davison, D. B., Page, R. D. M., Petsko, G. A., Stein, L. D., and Stormo, G. D., Eds.), pp. 6.5.1–6.5.14, John Wiley & Sons, Inc., New York
26. Madigan, D. M., and Raftery, A. E. (1994) Model selection and accounting for model uncertainty in graphical models using Occam's Window. *J Amer Stat Assoc* **89,** 1335–46.
27. Wasserman, L. (2000) Bayesian model selection and model averaging. *J Math Psychol* **44,** 92–107.
28. Hoeting, J. A., Madigan, D., and Raftery, A. E. (1999) Bayesian model averaging: a tutorial. *Stat Sci* **14,** 382–417.
29. Raftery, A. E. (1996) *Markov chain Monte Carlo in Practice* (Gilks, W. R., Richardson, S., and Spiegelhalter, D. J., Eds.), pp. 163–87, Chapman & Hall, London, New York.
30. Gilbert, D. (2007) ReadSeq, Indiana University, Bloomington.
31. Guindon, S., and Gascuel, O. (2003) A simple, fast, and accurate algorithm to estimate large phylogenies by maximum likelihood. *Syst Biol* **52,** 696–704.
32. Felsenstein, J. (2005) Phylip, Department of Genome Sciences, University of Washington, Seattle.
33. Gascuel, O. (1997) BIONJ: an improved version of the NJ algorithm based on a simple model of sequence data. *Mol Biol Evol* **14,** 685–95.
34. Swofford, D. L. (2000) PAUP*, Sinauer Associates, Sunderland, Massachusetts.
35. Bollback, J. P. (2002) Bayesian model adequacy and choice in phylogenetics. *Mol Biol Evol* **19,** 1171–80.
36. Ohta, T. (1992) Theoretical study of near neutrality. II. Effect of subdivided population structure with local extinction and recolonization. *Genetics* **130,** 917–23.
37. Goldman, N., and Whelan, S. (2000) Statistical tests of gamma-distributed rate heterogeneity in models of sequence evolution in phylogenetics. *Mol Biol Evol* **17,** 975–78.
38. Huelsenbeck, J. P., Larget, B., and Alfaro, M. E. (2004) Bayesian phylogenetic model selection using reversible jump Markov chain Monte Carlo. *Mol Biol Evol* **21,** 1123–33.
39. Jukes, T. H., and Cantor, C. R. (1969) *Mammalian Protein Metabolism* (Munro, H. M., Ed.), pp. 21–132, Academic Press, New York, NY.
40. Felsenstein, J. (1981) Evolutionary trees from DNA sequences: a maximum likelihood approach. *J Mol Evol* **17,** 368–76.
41. Kimura, M. (1980) A simple method for estimating evolutionary rate of base substitutions through comparative studies of nucleotide sequences. *J Mol Evol* **16,** 111–20.
42. Hasegawa, M., Kishino, K., and Yano, T. (1985) Dating the human-ape splitting by a molecular clock of mitochondrial DNA. *J Mol Evol* **22,** 160–74.
43. Tamura, K., and Nei, M. (1993) Estimation of the number of nucleotide substitutions in the control region of mitochondrial DNA in humans and chimpanzees. *Mol Biol Evol* **10,** 512–26.
44. Kimura, M. (1981) Estimation of evolutionary distances between homologous nucleotide sequences. *Proc Natl Acad Sci USA* **78,** 454–58.
45. Zharkikh, A. (1994) Estimation of evolutionary distances between nucleotide sequences. *J Mol Evol* **39,** 315–29.
46. Tavaré, S. (1986) *Some Mathematical Questions in Biology – DNA Sequence Analysis* (Miura, R. M., Ed.), Vol. 17, pp. 57–86, American Mathematical Society, Providence, RI.

Chapter 6

Estimating Maximum Likelihood Phylogenies with PhyML

Stéphane Guindon, Frédéric Delsuc, Jean-François Dufayard, and Olivier Gascuel

Abstract

Our understanding of the origins, the functions and/or the structures of biological sequences strongly depends on our ability to decipher the mechanisms of molecular evolution. These complex processes can be described through the comparison of homologous sequences in a phylogenetic framework. Moreover, phylogenetic inference provides sound statistical tools to exhibit the main features of molecular evolution from the analysis of actual sequences. This chapter focuses on phylogenetic tree estimation under the maximum likelihood (ML) principle. Phylogenies inferred under this probabilistic criterion are usually reliable and important biological hypotheses can be tested through the comparison of different models. Estimating ML phylogenies is computationally demanding, and careful examination of the results is warranted. This chapter focuses on PhyML, a software that implements recent ML phylogenetic methods and algorithms. We illustrate the strengths and pitfalls of this program through the analysis of a real data set. PhyML v3.0 is available from http://atgc_montpellier.fr/phyml/.

Key words: DNA and protein sequences, molecular evolution, sequence comparisons, phylogenetics, statistics, maximum likelihood, Markov models, algorithms, software, PhyML.

1. Introduction

In statistics, models are mathematical objects designed to approximate the processes that generated the data at hand. These models can be more or less complex, depending on the type of data and the available knowledge on the process that generated them. Each model has parameters which values need to be estimated from the data. Least-squares, maximum a posteriori estimation (MAP) or maximum likelihood (ML) are the main statistical frameworks suitable for this task. The least-squares criterion has been widely used in phylogenetics in order to build trees from matrices of pairwise distances between sequences. The last two criteria, ML

David Posada (ed.), *Bioinformatics for DNA Sequence Analysis,* Methods in Molecular Biology 537

DOI 10.1007/978-1-59745-251-9_6

and MAP, both rely on the probability that the data were generated according to the selected model. This conditional probability is the likelihood of the model. The next section gives a short description of the type of models that are used for phylogenetic inference.

1.1. Mathematical Description of Sequence Evolution

In molecular evolution, we commonly assume that homologous sequences evolve along a bifurcating tree or phylogeny. The topology of this tree describes the different clades or groups of taxa. The tree topology is generally considered to be the most important parameter of the whole phylogenetic model. The second parameter is the set of branch lengths on a given topology. The length of each branch on this topology represents an amount of evolution that corresponds to an expected number of nucleotide, codon or amino-acid substitutions. The last component of the model is a mathematical description of the process that generates substitutions during the course of evolution. Several assumptions are made about this process. We first consider that the sites of the alignment, i.e. its columns, evolve independently (*see* **Note 1**) and along the same phylogeny. We also assume that the substitution process is the same at the different sites of the alignment. This process is modelled with Markov chains. The two main components of a Markov model of substitution are (1) a symmetrical matrix that describes the relative speed at which the different substitution events occur (e.g. transition and transversion rates are generally distinct) and (2) the frequencies of the different states to be considered (i.e. nucleotides, codons or amino acids). The symmetrical matrix is often referred to as the exchangeability matrix. If combined with the vector of state frequencies, the resulting matrix corresponds to the generator of the Markov process and the vector of state frequencies is the stationary distribution.

1.2. A Difficult Optimisation Problem

Maximum likehood tree estimation consists of finding the phylogenetic model, i.e., the tree topology, branch lengths and parameters of the Markov model of substitution, that maximises the likelihood. Calculating the likelihood of a given phylogenetic model can be done efficiently using Felsenstein's pruning algorithm *(1)*. Unfortunately, finding the ML model is a difficult problem. The difficulty mostly comes from the very nature of the model itself. Indeed, while branch lengths and parameters of the substitution model are continuous variables, the tree topology is a discrete parameter. Hence, it is not surprising that, for most data sets, the likelihood function defines a rugged landscape with multiple peaks (see *(2)* however). Searching for the ML phylogenetic model in such conditions therefore rely on sophisticated optimisation methods, which combine both discrete and continuous optimisation procedures.

These methods are heuristics. As opposed to exact algorithms, heuristics do not guarantee to find the best (i.e. ML) solution. Despite this, simulation studies *(3,4)* suggest that the heuristics designed for phylogenetic estimation perform well overall. The most efficient methods are those that provide the best trade-off between their ability to maximise the likelihood function and the time spent to achieve this. Several methods/programs use deterministic approaches: given an initial non-optimal solution, the heuristic always follows the same path of intermediate solutions to reach the estimated ML one. As a consequence, given a particular input (i.e. data and model settings), these methods always produce the same output (i.e. the estimated ML phylogenetic model). Other approaches implement non-deterministic heuristics. These methods do not always follow the same path of intermediate solutions to reach the estimated ML tree. Hence, the same input potentially leads to distinct outputs (*see* **Note 2**).

1.3. Searching Through the Space of Tree Topologies

Another important distinction between tree building methods lies in the type of operations (or moves) that are used to explore the space of tree topologies. The three principal moves are nearest neighbour interchange (NNI), subtree pruning and regrafting (SPR) and tree bisection and reconnection (TBR) (see *(4)*). Each of these moves permits an exhaustive exploration of the space of tree topologies. However, the neighbourhood of trees defined by applying TBR moves to any given topology is much wider than the neighbourhood defined by SPR moves, which is itself larger than the neighbourhood defined by NNI moves. Therefore, TBRs are more efficient than SPRs or NNIs in jumping across very distinct tree topologies in just one step. The same also holds for SPR versus NNI moves. These differences result in various abilities to escape local maxima of the likelihood function. Consider a suboptimal peak of the likelihood surface and the tree topology found at this peak. Every potential NNI move will only allow to reach similar topologies. Therefore, such moves will sometimes fail to reach a higher peak of the likelihood surface. Hence, SPR operations are more efficient than NNIs in finding the highest peaks of the likelihood surface and TBR moves are more efficient than SPR ones. However, the large neighbourhood of trees defined by TBR compared to SPR or NNI also generally implies greater run times. Basically, many non-optimal solutions are evaluated when using TBR moves as compared to SPRs or NNIs. Hence, the best methods with respect to likelihood maximisation also tend to be the slowest ones.

1.4. PhyML v3.0: New Features

PhyML *(5)* is a software that estimates ML phylogenies from alignments of nucleotide or amino acid sequences. It provides a wide range of options that were designed to facilitate standard phylogenetic analyses. This chapter focuses on PhyML v3.0. This version of the program provides two important advances

compared to the previous releases. Indeed, PhyML v3.0 now proposes three different options to search across the space of phylogenetic tree topologies. It also implements a new method that evaluates branch supports. These new options are presented below.

1.5. Escaping Local Maxima

PhyML originally relies on a deterministic heuristic based on NNI moves. Traditional greedy approaches first evaluate the gain of likelihood brought by every possible NNI applied to the current topology. Only the best move is then used to improve the phylogeny at each step of the tree building process. Hence, most of the calculations evaluate moves that will not be used subsequently. To avoid such waste of potentially useful information, PhyML applies several "good" NNI moves simultaneously. Moreover, these local and simultaneous changes of the tree topology are accompanied by adjustments of every branch length in the phylogeny (*see* **Note 3**). Hence, most of the calculations that are performed during one step of the algorithm are actually used to improve the tree. This is what makes PhyML faster than popular algorithms such a fastDNAml *(6)*, and simulation results *(5)* demonstrated the accuracy of this approach. PhyML also outperforms the standard NNI-based greedy searching algorithms in terms of maximisation of the likelihood function. However, the analysis of real data and the comparison with SPR-based algorithms shows that PhyML occasionally gets trapped in local maxima. This shortcoming is more and more obvious as the number of sequences to analyse gets large (i.e. >50–100). Hence, while the phylogenetic models estimated with PhyML are generally good in terms of likelihoods, models with greater likelihoods can often be found.

This is the reason why the release 3.0 of PhyML proposes new SPR-based tree searching options. The methods implemented in PhyML v3.0 are inspired by Hordijk and Gascuel *(7)* work on the topic. Their approach essentially relies on using a fast distance-based method to filter out SPR moves that do not increase the likelihood of the current phylogenetic model. Hence, the likelihood function is only evaluated for the most promising moves. This strategy proves to be efficient in finding trees with high likelihoods. While the computational burden involved with SPRs is heavier than with simultaneous NNIs, this new approach is clearly less prone to be stuck in local maxima of the likelihood function (*see* **Note 4**).

PhyML 3.0 also proposes an intermediate option that includes both NNIs and SPRs. Simultaneous NNIs are first applied following the original algorithm, until no additional improvement is found. A single round of SPR moves are then tested: each subtree is pruned, regrafted, filtered and only the most promising moves are actually evaluated. If one or more SPR moves increase the likelihood, the best one is applied to the current tree. A new round of simultaneous NNIs then starts off after this step.

Simultaneous NNIs and SPRs therefore alternate until a maximum of the likelihood function is reached. This approach is generally faster than the SPR-only one but slower than the NNI-based heuristic. Also, while this strategy performs better than the NNI-only one in optimising the likelihood function, the SPR-only search often outperforms this mixed approach.

1.6. Fast Tests for Branch Support

PhyML v3.0 also provides users with a fast approximate likelihood ratio test (aLRT) for branches *(8)*, which proves to be a good alternative to the (time-consuming) bootstrap analysis. The aLRT is closely related to the conventional LRT, with the null hypothesis corresponding to the assumption that the tested branch has length 0. Standard LRT uses the test statistics $2(L_1 - L_0)$, where L_1 is the log-likelihood of the current tree, and L_0 the log-likelihood of the same tree, but with the branch of interest being collapsed. The aLRT approximates this test statistics in a slightly conservative but practical way as $2(L_1 - L_2)$, where L_2 corresponds to the second best NNI configuration around the branch of interest. Such a test is fast because the log-likelihood value L_2 is computed by optimising only over the branch of interest and the four adjacent branches, while other parameters are fixed at their optimal values corresponding to the best ML tree. Three branch supports computed from this aLRT statistics are available in PhyML v3.0: (1) the parametric branch support, computed from the χ^2 distribution (as usual with the LRT); (2) a non-parametric branch support based on a Shimodaira-Hasegawa-like procedure *(9)*; (3) a combination of these two supports, that is, the minimum value of both. The default is to use SH-like branch supports.

The rational behind the aLRT clearly differs from non-parametric bootstrap, as detailed in *(8)*. Basically, while aLRT values are derived from testing hypotheses, the bootstrap proportion is a repeatability measure; when the bootstrap proportion of a given clade is high, we are quite confident that this clade would be inferred again if another original data sample was available and analysed by the same tree-building method (which does not mean that the clade exists in the true tree). Also, computing aLRT values is much faster than getting bootstrap supports, as PhyML is run just once, while bootstrap requires launching PhyML 100–1,000 times. In fact, computing aLRT branch supports has a negligible computational cost in comparison with tree building. Note, however, that SH-like branch supports are non-parametric, just as are the bootstrap proportions. In fact, they often provide similar results as the bootstrap, pointing out the same poorly supported branches of the phylogeny, as illustrated in **Fig. 6.1**.

The aLRT assesses that the branch being studied provides a significant gain in likelihood, in comparison with the null hypothesis that involves collapsing that branch but leaving the rest of the tree topology identical. Thus, the aLRT does not account for other

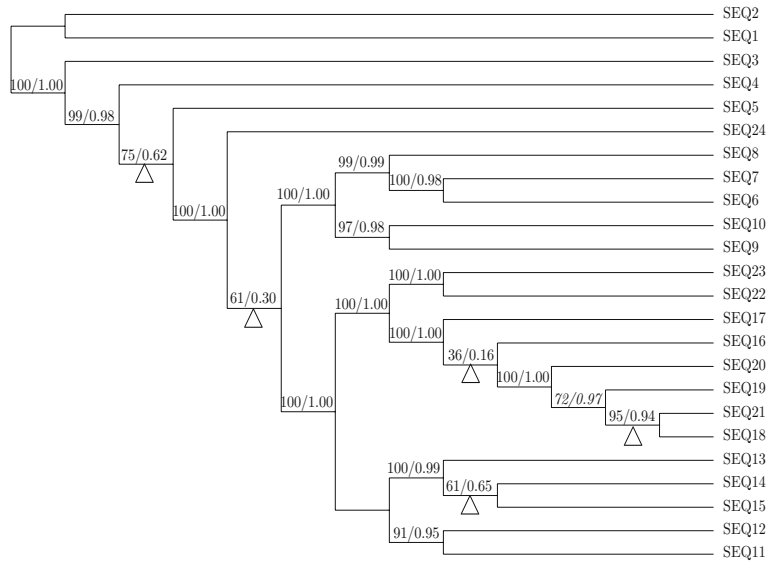


Fig. 6.1. Comparison of bootstrap and SH-like branch supports, using C8 alpha chain precursor data set from TREEBASE. This data set displays a strong phylogenetic signal overall as most internal branches (13/19) have support close to their maximum values with both approaches. SH-like supports are non-significant for five branches (noted with "△") that also have low bootstrap proportions, while one branch with a relatively low bootstrap proportion (72 in italic) is strongly supported according to the aLRT test (SH-like value : 0.97).

possible topologies that would be highly likely but quite different from the current topology. This implies that the aLRT performs well when the data contain a clear phylogenetic signal, but not as well as in the opposite case, where it tends to give a (too) local view on the branch of interest and be liberal. Note also that parametric χ^2 branch supports are based on the assumption that the evolutionary model used to infer the trees is the correct one. In that respect, the aLRT parametric interpretation is close to Bayesian posteriors. As the later, the resulting test is sometimes excessively liberal due to violations of the parametric assumptions.

Let us now focus on the practical aspects that go with ML phylogenetic model estimation using PhyML v3.0. The next section presents the inputs and outputs of the program, the different options and how to use them.

2. Program Usage

PhyML v3.0 has two different user interfaces. The default is to use the PHYLIP-like text interface (**Fig. 6.2**) by simply typing "phyml" in a command-line window or by clicking on the PhyML icon (*see*

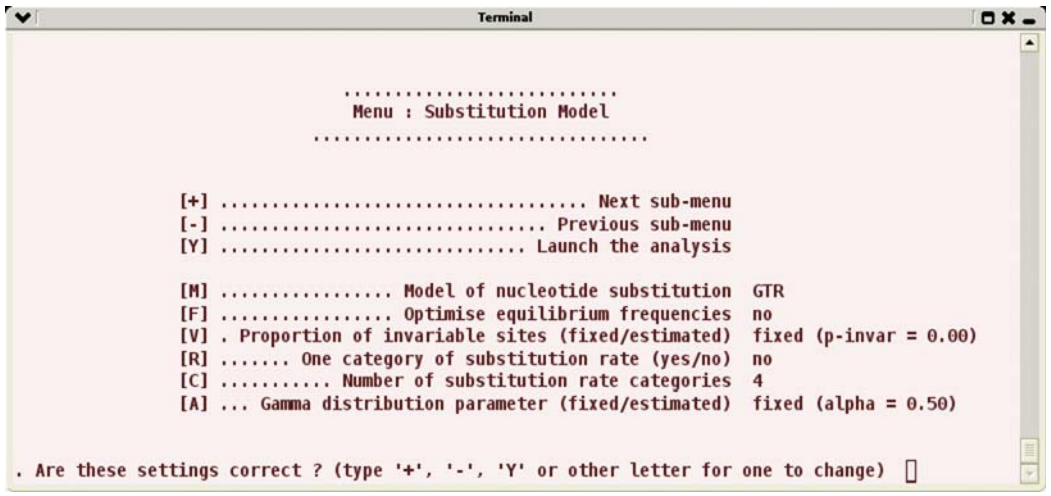


Fig. 6.2. Text-based interface to PhyML. The PHYLIP-like text interface to the program is organised in sub-menus. The "+" and "-" keys are used to cycle through them. For each sub-menu, the defaults options can be altered by entering the relevant keys. The data analysis is launched once the "y" (or "Y") key is entered.

Note 5). After entering the name of the input sequence file, the user goes through a list of sub-menus that allow her/him to set up the analysis. There are currently four distinct sub-menus:

1. *Input Data*: specify whether the input file contains amino acid or nucleotide sequences. What is the sequence format (*see* **Section 2.1**) and how many data sets should be analysed.
2. *Substitution Model*: selection of the Markov model of substitution (*see* **Section 2.2**).
3. *Tree Searching*: selection of the tree topology searching algorithm (*see* **Section 2.4**).
4. *Branch Support*: selection of the method that is used to measure branch support (*see* **Section 2.3**).

"+" and "-" keys are used to move forward and backward in the sub-menu list. Once the model parameters have been defined, typing "Y" (or "y") launches the calculations. The meaning of some options may not be obvious to users who are not familiar with phylogenetics. In such situation, we strongly recommend to use the default options. As long as the format of the input sequence file is correctly specified (sub-menu *Input data*), the safest option for non-expert users is to use the default settings.

The alternative to the PHYLIP-like interface is the command line. Users that do not need to modify the default parameters can launch the program with the "phyml -i your_input_sequence_file_name" command. The list of all command line arguments and how to use them is given in the "Help" section, which is displayed after entering the "phyml help" command. Command lines are specially handy for launching PhyML v3.0 in batch mode. Note however

that some options are only available through the PHYLIP-like interface. Hence, options that are not listed in the "Help" section may be accessible through the interactive text interface.

2.1. Inputs / Outputs

PhyML reads data from standard text files, without the need for any particular file name extension. Alignments of DNA or protein sequences must be in PHYLIP sequential or interleaved format (see **Fig. 6.3a**). The first line of the input file contains the number of species and the number of characters, in free format, separated by blanks. One slight difference with PHYLIP format concerns sequence name lengths. While PHYLIP format limits this length to ten characters, PhyML can read up to hundred-character-long sequence names. Blanks and the symbols "(),:" are not allowed within sequence names because the NEWICK tree format makes special use of these symbols (**Fig. 6.3b**).

Another slight difference with PHYLIP format is that actual sequences must be separated from their names by at least one blank character. These sequences must not be longer than 10^6 amino acid or nucleotide characters and a given data set can have up to 4×10^3 of them. However, the size of the largest data set PhyML v3.0 can process depends on the amount of physical memory available. To avoid overflows, PhyML v3.0 pauses when the estimated amount of memory that needs to be allocated exceeds 250 Mb. The user can then decide whether she/he wants to continue or cancel the analysis (*see* **Note 6**).

a)

```
5 60
first_seq_name  CCATCTCACGGTCGGTACGATACACCKGCTTTTGGCAGGAAATGGTCAATATTACAAGGT
second_seq_name CCATCTCACGGTCAG---GATACACCKGCTTTTGGCGGGAAATGGTCAACATTAAAAGAT
third_seq_name  RCATCTCCCGCTCAG---GATACCCCKGCTGTTG????????????????ATTAAAAGGT
fourth_seq_name RCATCTCATGGTCAA---GATACTCCTGCTTTTGGCGGGAAATGGTCAATCTTAAAAGGT
fifth_seq_name  RCATCTCACGGTCGGTAAGATACACCTGCTTTTGGCGGGAAATGGTCAAT????????GT

5 40
first_seq_name  CCATCTCANNNNNNNNACGATACACCKGCTTTTGGCAGG
second_seq_name CCATCTCANNNNNNNNNGGGATACACCKGCTTTTGGCGGG
third_seq_name  RCATCTCCCGCTCAGTGAGATACCCCKGCTGTTGXXXXX
fourth_seq_name RCATCTCATGGTCAATG-AATACTCCTGCTTTTGXXXXX
fifth_seq_name  RCATCTCACGGTCGGTAAGATACACCTGCTTTTGxxxxx
```

b)

```
((first_seq_name:0.03,second_seq_name:0.01):0.04,third_seq_name:0.01,(fourth_seq_name:0.2,fifth_seq_name:0.05));
((third_seq_name:0.04,second_seq_name:0.07):0.02,first_seq_name:0.02,(fourth_seq_name:0.1,fifth_seq_name:0.06));
```

Fig. 6.3. Typical DNA sequence alignments (a) and input trees (b). Sequence names do not contain any blank character and at least one blank separates each name from the corresponding sequence. Trees are in standard NEWICK format.

An input sequence file may also display more than a single data set. Each of these data sets must be in PHYLIP format and two successive alignments must be separated by an empty line. Processing multiple data sets requires to toggle the "M" option in the *Input Data* sub-menu or use the "–n" command line option and enter the number of data sets to analyse. The multiple data set option can be used to process re-sampled data that were generated using a non-parametric procedure such as cross-validation or jackknife (a bootstrap option is already included in PhyML). This option is also useful in multiple gene studies, even if fitting the same substitution model to all data sets may not be suitable.

Gaps correspond to the "–" symbol. They are systematically treated as unknown characters "on the grounds that we don't know what would be there if something were there" (J. Felsenstein, PHYLIP main documentation). The likelihood at these sites is summed over all the possible states (i.e. nucleotides or amino acids) that could actually be observed at these particular positions. Note however that columns of the alignment that display only gaps or unknown characters are simply discarded because they do not carry any phylogenetic information (they are equally well explained by any model). PhyML v3.0 also handles ambiguous characters such as *R* for *A* or *G* (purines) and *Y* for *C* or *T* (pyrimidines). **Tables 6.1** and **6.2** give the list of valid characters/symbols and the corresponding nucleotides or amino acids.

PhyML v3.0 can read one or several phylogenetic trees from an input file. This option is accessible through the *Tree Searching* sub-menu or the "–u" argument from the command line. Input

Table 6.1
List of valid characters in DNA sequences and the corresponding nucleotides

Character	Nucleotide	Character	Nucleotide
A	Adenosine	*Y*	*C* or *T*
G	Guanine	*K*	*G* or *T*
C	Cytosine	*B*	*C* or *G* or *T*
T	Thymine	*D*	*A* or *G* or *T*
U	Uracil (=*T*)	*H*	*A* or *C* or *T*
M	*A* or *C*	*V*	*A* or *C* or *G*
R	*A* or *G*	– or *N* or *X* or ?	Unknown
W	*A* or *T*		(=*A* or *C* or *G* or *T*)
S	*C* or *G*		

Table 6.2
List of valid characters in protein sequences and the corresponding amino acids

Character	Amino Acid	Character	Amino Acid
A	Alanine	*L*	Leucine
R	Arginine	*K*	Lysine
N or *B*	Asparagine	*M*	Methionine
D	Aspartic acid	*F*	Phenylalanine
C	Cysteine	*P*	Proline
Q or *Z*	Glutamine	*S*	Serine
E	Glutamic acid	*T*	Threonine
G	Glycine	*W*	Tryptophan
H	Histidine	*Y*	Tyrosine
I	Isoleucine	*V*	Valine
L	Leucine	– or *X* or ?	Unknown
K	Lysine		(can be any amino acid)

trees are generally used as initial ML estimates to be subsequently adjusted by the tree-searching algorithm (*see* **Section 2.4**). This option is also helpful when one wants to evaluate the likelihood on a particular set of (possibly competing) phylogenetic trees. Trees should be in standard NEWICK format (**Fig. 6.3b**). They can be either rooted or unrooted and multifurcations are allowed. Taxa names must, of course, match the corresponding sequence names.

Single or multiple sequence data sets may be used in combination with single or multiple input trees. When the number of data sets is one ($n_D = 1$) and there is only one input tree ($n_T = 1$), then this tree is simply used as input for the single data set analysis. When $n_D = 1$ and $n_T > 1$, each input tree is used successively for the analysis of the single alignment. If $n_D > 1$ and $n_T = 1$, the same input tree is used for the analysis of each data set. The last combination is $n_D > 1$ and $n_T > 1$. In this situation, the i-th tree in the input tree file is used to analyse the i-th data set. Hence, n_D and n_T must be equal here.

Table 6.3 presents the list of files resulting from a PhyML v3.0 analysis. Basically, each output file name can be divided into three parts. The first part is the sequence file name, the second part corresponds to the extension "_phyml_" and the third part is related to the file content. When launched with the default options, PhyML v3.0 only creates two files: the tree file and the

Table 6.3
Standard output files

Output file name	Content
seq_phyml_tree.txt	ML tree
seq_phyml_stats.txt	ML model parameters
seq_phyml_boot_trees.txt	ML trees – bootstrap replicates
seq_phyml_boot_stats.txt	ML model parameters – bootstrap replicates

Sequence file name : "seq"

model parameter file. The estimated ML tree is in standard NEWICK format (**Fig. 6.3b**). The model parameters file, or statistics file, displays the ML estimates of the substitution model parameters, the likelihood of the ML phylogenetic model and other important information concerning the settings of the analysis (e.g. type of data, name of the substitution model, starting tree (*see* **Section 2.4**), etc.). Two additional output files are created if bootstrap supports were evaluated. These files simply contain the ML trees and the substitution model parameters estimated from each bootstrap replicate. Such information can be used to estimate sampling errors around each parameter of the phylogenetic model. The best ML tree file is only created when the estimation of the phylogeny resulted from multiple random starting trees. It contains the tree with the highest likelihood that was found among all the trees estimated during the analysis (*see* **Section 2.4**).

2.2. Substitution Models

PhyML implements a wide range of substitution models: JC69 *(10)*, K80 *(11)*, F81 *(1)*, F84 *(12)*, HKY85 *(13)*, TN93 *(14)* GTR *(15,16)* and CUSTOM for nucleotides; WAG *(17)*, Dayhoff *(18)*, JTT *(19)*, Blosum62 *(20)*, mtREV *(21)*, rtREV *(22)*, cpREV *(23)*, DCMut *(24)*, VT *(25)* and mtMAM *(26)* for amino acids. Nucleotide equilibrium frequencies are estimated by counting the occurrence of *A*, *C*, *G* and *T*s in the data (*see* **Note 7**). These frequencies can also be adjusted to maximise the likelihood of the phylogenetic model (*Substitution Model* sub-menu, option "F" ; command line argument : "–f e"). Amino acid equilibrium frequencies are either deduced from the actual sequences (*Substitution Model* sub-menu, option "F"; command line argument: "–f e") or given by the substitution models themselves (default option).

The CUSTOM option provides the most flexible way to specify the nucleotide substitution model. The model is defined by a string made of six digits. The default string is "000000", which means that the six relative rates of nucleotide changes: $A \leftrightarrow C$, $A \leftrightarrow G$, $A \leftrightarrow T$, $C \leftrightarrow G$, $C \leftrightarrow T$ and $G \leftrightarrow T$ are equal. The

string "010010" indicates that the rates $A \leftrightarrow G$ and $C \leftrightarrow T$ are equal and distinct from $A \leftrightarrow C = A \leftrightarrow T = C \leftrightarrow G = G \leftrightarrow T$. This model corresponds to HKY85 (default) or K80 if the nucleotide frequencies are all set to 0.25. "010020" and "012345" correspond to TN93 and GTR models, respectively. The digit string therefore defines groups of relative substitution rates. The initial rate within each group is set to 1.0, which corresponds to F81 (JC69 if the base frequencies are equal). Users also have the opportunity to define their own initial rate values (this option is only available through the PHYLIP-like interface). These rates are then optimised afterwards (option "O") or fixed to their initial values. The CUSTOM option can then be used to implement all substitution models that are special cases of GTR.

PhyML v3.0 also implements Ziheng Yang's discrete gamma model *(27)* to describe the variability of substitution rates across nucleotide or amino acids positions. Users can specify the number of substitution-rate categories (*Substitution Model* sub-menu option "C" or "-c" argument from the command line) and choose to estimate the gamma shape parameter from the data or fix its value *a priori* (option "A" or "-a"). The program also handles invariable sites (option "V" or "-v"). Here again, the value of this parameter can be estimated in the ML framework or fixed *a priori* by the user (*see* **Note 8**)

2.3. Branch Support

PhyML v3.0 proposes two main options to assess the support of the data for non-terminal branches in the phylogeny. The most popular approach relies on non-parametric bootstrap. This option is available through the PHYLIP-like interface (*Branch Support* sub-menu) or the "-b" argument from the command line. Users only have to specify the number of replicates that will be generated to work out the bootstrap values. The ML output tree (*see* **Section 2.1**) will then display both branch lengths and bootstrap values. It is very important to keep in mind that bootstrap values are displayed on the ML tree estimated from the original data set. There is no consensus tree reconstruction involved here. Note however that PhyML also outputs every tree estimated from the re-sampled data sets. These trees can then be used to build a consensus tree, using the program CONSENSE from the PHYLIP package for instance. However, there is no guarantee that this consensus tree is also the ML tree (but both generally have similar topologies).

PhyML v3.0 also implements the aLRT described in **Section 1.6**. Just like bootstrap support, aLRT options are available from the *Branch Support* sub-menu in the PHYLIP-like interface. The "A" key is used to choose among four distinct options as aLRT supports can be assessed through different tests. The default is to use SH-like branch supports. It is also possible to test aLRT values against a χ^2 distribution or to use a combination of these last two options. Expert users also have the opportunity to retrieve the

aLRT values themselves in order to examine the differences of likelihood among competing topologies. These four options are also available using the "-b" argument from the command line as explained in the "Help" section.

2.4. Tree-Searching Algorithms

PhyML v3.0 implements three heuristics to explore the space of tree topologies (*see* **Section 1.4**). All methods take as input a starting tree that will be improved subsequently. The default is to let PhyML v3.0 build this starting tree using BioNJ *(28)*. However, users can also give their own input tree(s) in NEWICK format (*see* **Section 2.1**). Under the default settings, the starting tree is improved using simultaneous NNIs. PhyML v3.0 also proposes NNI+SPR and full-SPR heuristics that both search more thoroughly the space of tree topologies. Simultaneous NNIs, NNI+SPR and full-SPR options are available from the *Tree Searching* sub-menu or the "-s" argument from the command line.

Full-SPR searches can also be combined with multiple random starting trees (*Tree Searching* sub-menu, options "S" and "R"). Multiple random starting solutions are useful to evaluate the difficulty of the optimisation problem given the data at hand. Indeed, if the estimated solution (the ML phylogenetic model in our case) strongly depends on the value that is used to initiate the optimisation process (the starting tree), then the function to optimise is probably not smooth and local optima are likely to be commonplace. Hence, multiple random starting trees provide a diagnosis tool to evaluate the smoothness of the likelihood surface and the robustness of the inferred tree. The default is to use five random starting trees. Five ML phylogenetic models are then estimated and the best phylogenetic tree (i.e. the tree with the highest likelihood) is printed in the "phyml_tree.txt" file (**Table 6.3**). When all these trees are identical, this tree is likely to be the ML solution.

2.5. Recommendations on Program Usage

From the user perspective, the choice of the tree-searching algorithm among those provided by PhyML v3.0 is probably the toughest one. The fastest option relies on local and simultaneous modifications of the phylogeny using NNI moves. More thorough explorations of the space of topologies are also available through the NNI+SPR and full-SPR options. As these two classes of tree-topology moves involve different amounts of computation, it is important to determine which option is the most suitable for the type of data set or analysis one wants to perform. Below is a list of recommendations for typical phylogenetic analyses.

1. *Single data set, unlimited computing time*. The best option here is probably to use a full-SPR search. If the focus is on estimating the relationships between species (*see* **Section 3**), it is a good idea to use more than one starting tree to decrease the chance of getting stuck in a local maximum of the

likelihood function. Note however that NNI+SPR or NNI are also appropriate if the analysis does not mainly focus on estimating the evolutionary relationships between species (e.g. a tree is needed to estimate the parameters of codon-based models later on). Branch supports can be estimated using bootstrap and aLRT.

2. *Single data set, restricted computing time.* The three tree-searching options can be used depending on the computing time available and the size of the data set. For small data sets (i.e. < 50 sequences), NNI will generally perform well provided that the phylogenetic signal is strong. It is relevant to estimate a first tree using NNI moves and examine the reconstructed phylogeny in order to have a rough idea of the strength of the phylogenetic signal (the presence of small internal branch lengths is generally considered as a sign of a weak phylogenetic signal, specially when sequences are short). For larger data sets (> 50 sequences), full-SPR search is recommended if there are good evidence of a lack of phylogenetic signal. Bootstrap analysis will generally involve large computational burdens. aLRT branch supports therefore provide an interesting alternative here.
3. *Multiple data sets, unlimited computing time.* Comparative genomic analyses sometimes rely on building phylogenies from the analysis of a large number of gene families. Here again, the NNI option is the most relevant if the focus is not on recovering the most accurate picture of the evolutionary relationships between species. More time-consuming heuristics (NNI+SPR and full-SPR) should be used when the topology of the tree is an important parameter of the analysis (e.g. identification of horizontally transferred genes using phylogenetic tree comparisons). Internal branch support is generally not a crucial parameter of the multiple data set analyses. Using aLRT statistics is therefore the best choice.
4. *Multiple data sets, limited computing time.* The large amount of data to be processed in a limited time generally requires the use of the fastest tree searching and branch support estimation methods. Hence, NNI and aLRT are generally the most appropriate here.

Another important point is the choice of the substitution model. While default options generally provide acceptable results, it is often warranted to perform a pre-analysis to identify the best-fit substitution model. This pre-analysis can be done using popular software such as Modeltest *(29)* or ProtTest *(30)*, for instance. These programs generally recommend the use of a discrete gamma distribution to model the substitution process as variability of rates among sites is a common feature of molecular evolution. The choice of the number of rate classes to use for this distribution is also an

important one. While the default is set to four categories in PhyML v3.0, it is recommended to use larger number of classes if possible in order to best approximate the patterns of rate variation across sites *(31)*. Note however that run times are directly proportional to the number of classes of the discrete gamma distribution. Here again, a pre-analysis with the simplest model should help the user to determine the number of rate classes that represents the best trade-off between computing time and fit of the model to the data.

3. Example

As an illustration of using PhyML v3.0, we focus on reconstructing placental mammal phylogeny from the comparison of complete mitochondrial genomes. In such a situation, taxonomic purposes are clearly more important than deciphering the evolutionary processes. This particular class of problems usually involves huge computational burdens. The next sections present the data and go through the different steps of the analysis.

3.1. The Data

Reconstructing the evolutionary history of major placental mammal lineages has been a long-standing phylogenetic challenge. For long restricted to the study of morphological characters, placental mammal phylogenetics has widely benefited from the advent of molecular techniques giving access to a large number of informative characters from both mitochondrial and nuclear genomes. After a period of relative confusion mainly due to restricted taxon and gene sampling, phylogenetic analyses of placental mammal relationships based on complete mitochondrial genomes *(32,33)* and concatenated nuclear genes *(34–36)* have converged towards congruent solutions in striking contrast to morpho-anatomical data. According to these new phylogenies, four major groups of placental mammals have been recognised: (1) Afrotheria also known as the African clade (Elephants, Sirenians, Hyraxes, Aardvark, Elephant Shrews, Golden Moles and Tenrecs), (2) Xenarthra (Armadillos, Anteaters and Sloths), a clade of South American endemics, and two distinct groups that comprise most of today's mammalian diversity with (3) Euarchontoglires (Lagomorphs, Rodents, Tree Shrews, Flying Lemurs and Primates) and (4) Laurasiatheria (Insectivores, Bats, Pangolins, Carnivorans, Perissodactyls, Artiodactyls and Cetaceans).

As stated above, phylogenetic reconstruction of deep placental mammal relationships based on mitochondrial genomes had first been hampered by the peculiar evolutionary properties of this molecule. Indeed, the mitochondrial genome evolving about four times more rapidly than the nuclear genome, the recovery of

the earliest divergences *(37)* proved to be difficult due to substitutional saturation. In particular, the first analyses of complete mitochondrial genomes created a controversy surrounding the origin of rodents that were initially found to be paraphyletic *(38)*. Indeed, murid rodents (mice and rats) emerged first among placentals in most mitogenomic trees *(39,40)*. This unexpected finding was actually the result of a long-branch attraction artifact due to the fast evolving murids *(41,42,32)*. Thus, for illustrative purposes, we assembled a complete mitochondrial genome data set from 38 species representing all placental orders (**Table 6.4**). The concatenation of the 12 H-stranded mitochondrial protein-coding genes (NADH Dehydrogenase subunits 1, 2, 3, 4, 4L and 5; Cytochrome c Oxydase subunits I, II and III, ATP Synthase F0 subunits 6 and 8; and Cytochrome b) led to an alignment of 3,507 amino acid sites after removing ambiguously aligned positions.

Table 6.4
List of the mammalian species and the corresponding GenBank accession numbers of the complete mitochondrial genomes that were analysed in this chapter (*see* Section 3.1)

Species	Common name	Accession number
Ornithorhynchus anatinus	Platypus	NC_000891
Macropus robustus	Hill Wallaroo	NC_001794
Didelphis virginiana	Virginia Opossum	NC_001610
Monodelphis domestica	Gray Short-tailed Opossum	NC_006299
Dugong dugon	Dugong	NC_003314
Loxodonta africana	African Elephant	NC_000934
Procavia capensis	Rock Hyrax	NC_004919
Orycteropus afer	Aardvark	NC_002078
Chrysochloris asiatica	Cape Golden Mole	NC_004920
Echinops telfairi	Lesser Hedgehog Tenrec	NC_002631
Elephantulus sp.	Elephant Shrew	NC_004921
Macroscelides proboscideus	Short-eared Elephant Shrew	NC_004026
Dasypus novemcinctus	Nine-banded Armadillo	NC_001821
Choloepus didactylus	Southern Two-toed Sloth	NC_006924

(continued)

Table 6.4 (continued)

Species	Common name	Accession number
Tamandua tetradactyla	Southern Tamandua	NC_004032
Homo sapiens	Human	NC_001807
Lemur catta	Ring-tailed Lemur	NC_004025
Lepus europaeus	European Hare	NC_004028
Ochotona collaris	Collared Pika	NC_003033
Sciurus vulgaris	Eurasian Red Squirrel	NC_002369
Cavia porcellus	Guinea Pig	NC_000884
Mus musculus	House Mouse	NC_005089
Rattus norvegicus	Brown Rat	NC_001665
Sorex unguiculatus	Long-clawed Shrew	NC_005435
Talpa europaea	European Mole	NC_002391
Artibeus jamaicensis	Jamaican Fruit-eating Bat	NC_002009
Pteropus dasymallus	Ryukyu Flying Fox	NC_002612
Canis familiaris	Dog	NC_002008
Felis catus	Cat	NC_001700
Equus caballus	Horse	NC_001640
Ceratotherium simum	White Rhinoceros	NC_001808
Tapirus terrestris	Brazilian Tapir	NC_005130
Lama pacos	Alpaca	NC_002504
Sus scrofa	Pig	NC_000845
Bos taurus	Cattle	NC_006853
Hippopotamus amphibius	Hippopotamus	NC_000889
Balaenoptera physalus	Finback Whale	NC_001321
Lagenorhynchus albirostris	White-beaked Dolphin	NC_005278

3.2. NNI Heuristic Search

A ML phylogenetic model was first estimated using simultaneous NNI moves to improve a BioNJ starting tree (default option) under the mtMAM substitution model. The shape parameter (α) of a gamma distribution with four bins as well as the proportion of invariants (p-inv) were both estimated from the data. All together, the substitution model is noted mtMAM+Γ4+I. The estimated phylogeny is presented in **Fig. 6.4**.

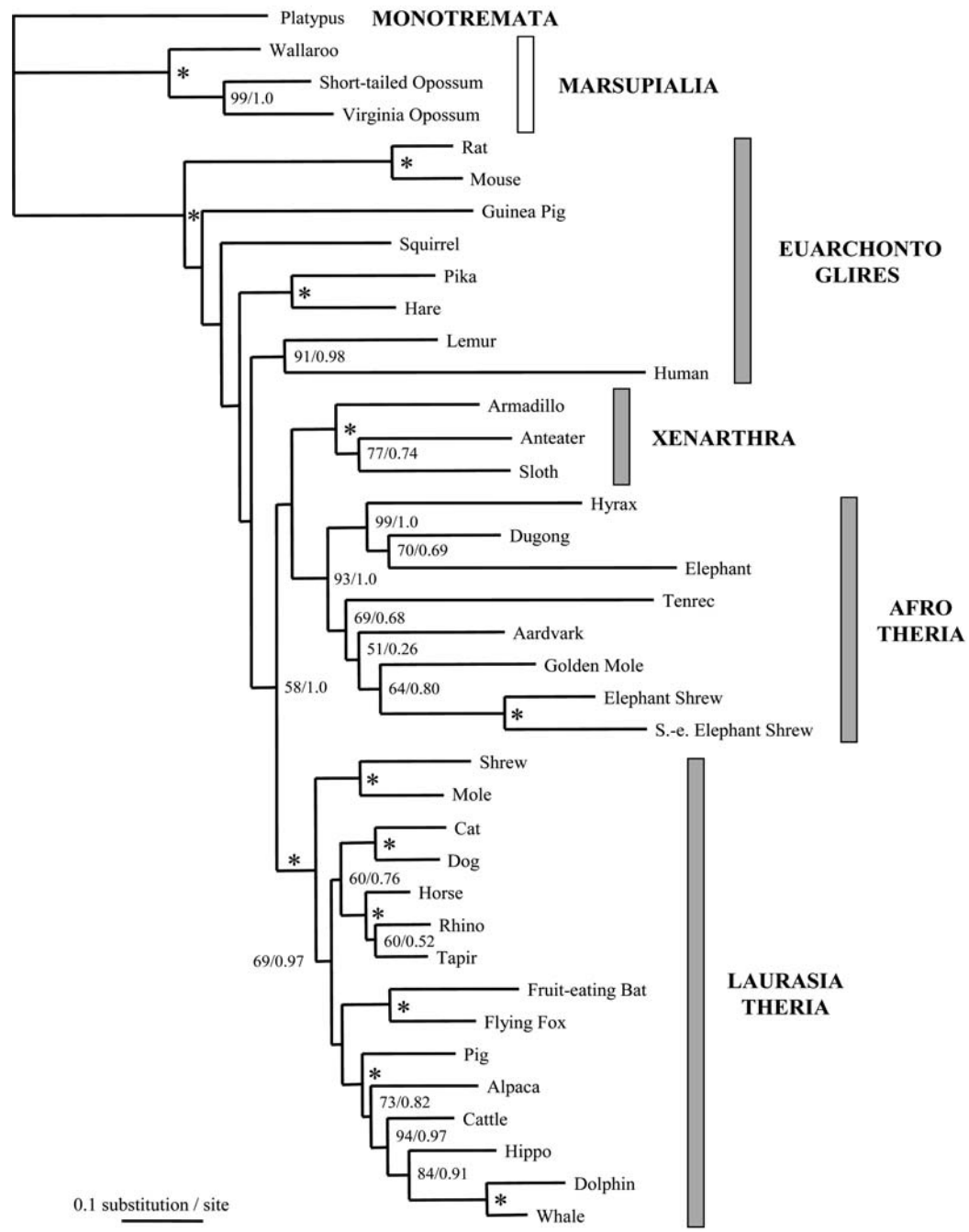


Fig. 6.4. ML phylogeny reconstructed using simultaneous NNI moves. The log-likelihood of the corresponding phylogenetic model is −74040.38. Numbers in the tree correspond to non-parametric bootstrap supports (100 replicates) and *p*-values of the approximate likelihood ratios (SH-test). Values are reported only for nodes with BP $>$ 50 and stars indicate nodes that received 100% support. S.-e. Elephant Shrew is for Short-eared Elephant Shrew.

This analysis, including the calculation of aLRT-based branch supports, took about 25 min to complete on an AMD Opteron 250 2.4 Ghz processor running Linux. The estimated model parameters ($\alpha = 0.69$; p-inv = 0.34) indicates strong among site rate heterogeneity within the concatenation. The log likelihood of the whole phylogenetic model is -74040.38. Interestingly, three out of the four recognised major placental clades are recovered as monophyletic: Afrotheria, Xenarthra and Laurasiatheria. The fourth group, Euarchontoglires, appears paraphyletic at the base of the tree with murid rodents (Mouse and Rat) emerging first and successively followed by Guinea Pig, Squirrel, Lagomorphs (Hare and Pika) and Primates (Lemur and Human). In fact, this topology differs from what is expected to be the most accurate placental phylogeny because of the position of the murid rodent clade. This peculiar topology is reminiscent of early mitogenomic studies. It is likely to be the result of a long-branch attraction artifact that is caused by the high evolutionary rate of murid mitochondrial genomes relative to other rodents. Worth noting, however, is the fact that the BioNJ starting topology also presents this apparent rooting artifact (not shown). This observation suggests a potential influence of the starting tree on the outcome of the NNI heuristic search. To check whether the likely artefactual topology found in **Fig. 6.4** is a consequence of using the BioNJ topology as a starting tree for the NNI heuristic search, we conducted the same analysis using a maximum parsimony (MP) tree as the starting topology. Using this strategy results in a slightly better model with respect to the likelihood (-74040.25). The corresponding ML topology only differs from the one in **Fig. 6.4** in the position of Bats (Fruit-eating Bat and Flying-Fox) that now emerge second within Laurasiatheria (not shown). However, as the MP topology also suffers from the rooting artifact, it is still possible that the NNI heuristic search from this starting tree may have reached a local maximum of the likelihood function.

3.3. SPR Heuristic Search

The ML phylogeny reconstructed by applying full-SPR moves to the BioNJ starting tree under the mtMAM+Γ4+I model is presented in **Fig. 6.5**. Building the tree took about 1 h 25 min to complete on the same computer, which is almost exactly 1 h more than the search using NNI moves. As previously, aLRT branch supports were also calculated and the time needed to work out these values was approximately 5% of the total computing time. The estimated model parameters ($\alpha = 0.68$; p-inv $= 0.34$) are almost identical to the ones obtained using the NNI search strategy. This result confirms that ML estimation of model parameters is relatively insensitive to topological differences induced by the different heuristics. However, SPR moves converged to a model that has a greater log likelihood (-74021.74) than the NNI-based one (-74040.38). The SPR topology presents major differences with the NNI one, since the respective monophyly of each of the

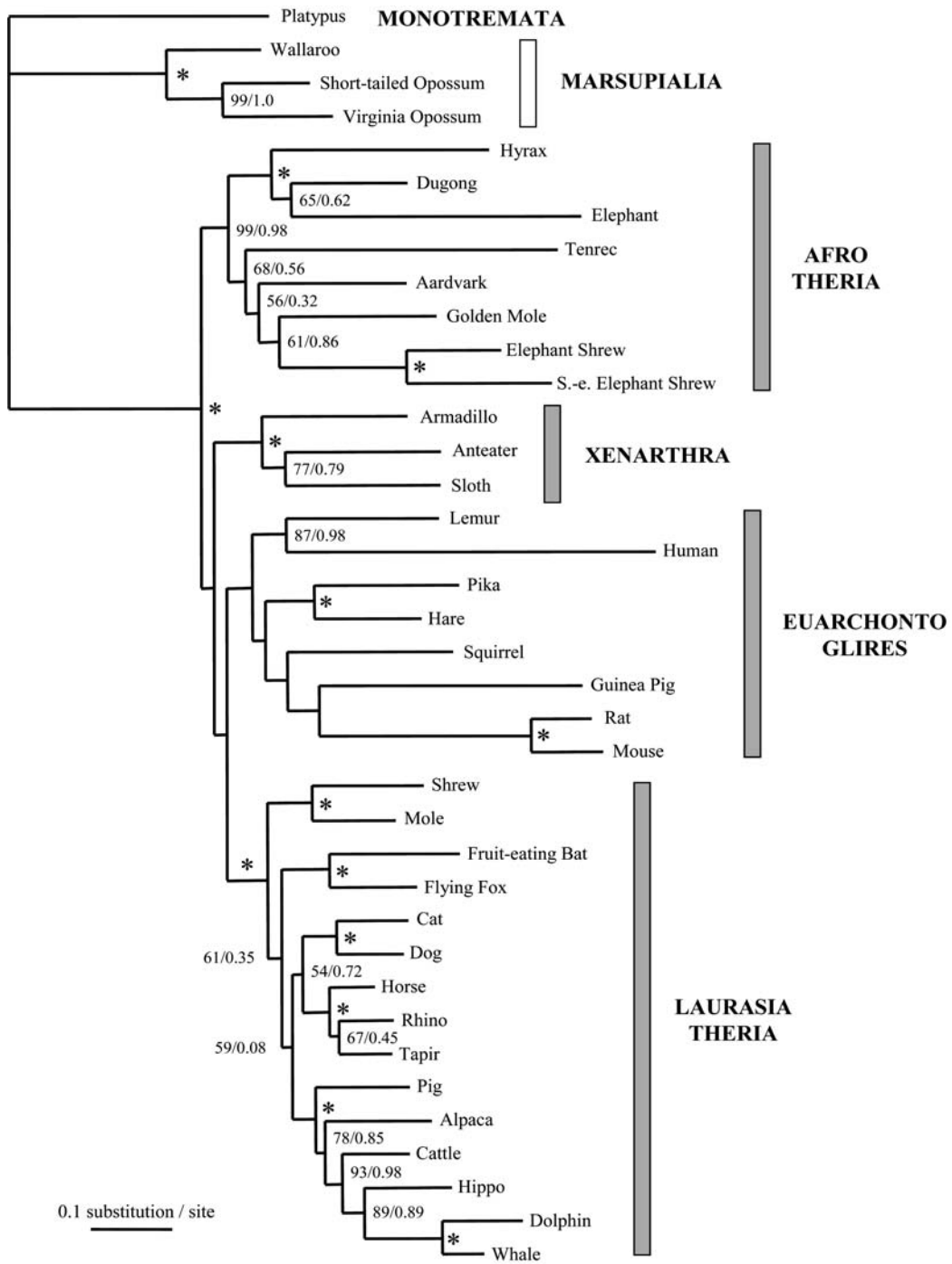


Fig. 6.5. ML phylogeny reconstructed using SPR moves. The log-likelihood of the corresponding phylogenetic model is −74021.74. Numbers in the tree correspond to non-parametric bootstrap supports (100 replicates) and *p*-values of the approximate likelihood ratios (SH-test). Values are reported only for nodes with BP $>$ 50 and stars indicate nodes that received 100% support. S.-e. Elephant Shrew is for Short-eared Elephant Shrew.

four major placental clades is now recovered (**Fig. 6.5**). Indeed, the ML topology is no longer rooted on the Mouse/Rat ancestral branch with Euarchontoglires being monophyletic and Afrotheria now appearing as the earliest placental offshoot. This topology is almost fully compatible with the new placental phylogeny inferred from large concatenated data sets of mainly nuclear genes *(34–36)*. This result illustrates that, in this particular case, NNI-based heuristic is trapped in a local maximum because of insufficient tree space exploration conditioned by the BioNJ or MP starting topologies. Note however that the NNI-based and SPR-based ML phylogenetic models are not statistically different according to a SH test *(9)*. Hence, while simultaneous NNIs get trapped in a local maximum, the estimated ML solution is not different from the one found by SPRs from a statistical point of view. In other words, NNI tree searching is probably trapped in this local optimum because of the lack of phylogenetic signal for certain parts of the tree.

3.4. SPR Heuristic Search Starting from Random Trees

SPR searches can also be initiated with random starting trees (*see* **Section 2.4**). The full-SPR-based tree search procedure was thus repeated ten times, each analysis starting from a different random phylogeny. In eight of these ten replicates, the ML topology of **Fig. 6.5** compatible with the new placental phylogeny was recovered (log-likelihood: −74021.74). However, in the two remaining replicates (20%), the SPR heuristic search converged to the alternative suboptimal topology (log-likelihood: −74040.25) that was previously found when using a MP starting tree combined with the NNI heuristic search (*see* **Section 3.2**). These results suggest that the likelihood surface for this placental mitogenomic data set is dominated by at least two peaks relatively close in likelihood but distant in the space of tree topologies, one probably corresponding to the ML topology and the other to the suboptimal topology caused by the rooting artifact.

3.5. Assessing Statistical Support for Internal Edges

Statistical supports for branches displayed by each of the two competing topologies were measured using 100 non-parametric bootstrap replicates (BP; *(43)*) and the approximate likelihood ratio test for branches (aLRT; *(8)*). BP and aLRT values (more precisely, *p*-values obtained from SH tests) are reported on **Figs. 6.4** and **6.5** for nodes that show at least 50% bootstrap support. For both topologies, the statistical support is almost maximal for the respective monophyly of Afrotheria, Xenarthra and Laurasiatheria. The monophyly of Euarchontoglires recovered in the SPR topology (**Fig. 6.5**) is not statistically supported, as is also its paraphyly induced by the rooting artifact in the NNI topology (**Fig. 6.4**). In fact, the differences between the two topologies only involve nodes with weak support from the data.

Bootstrap and aLRT values largely agree on most branches of both trees. Indeed, branches with bootstrap supports close to 100

also have aLRT values close to 1.0 in most cases. However, a few branches are well supported according to aLRT values but have small bootstrap proportions (see the branch at the root of the Xenarthra, Afrotheria and Laurasiatheria clades, with BP=58 and aLRT=1.0). Such differences between aLRT and bootstrap values are probably the consequence of a lack of phylogenetic signal to resolve specific parts of the phylogeny. More work still needs to be done to fully understand such results though.

4. Notes

1. The hypothesis of site independence is relaxed in certain models. For instance, codon-based models impose a constraint on groups of columns that belong to the same codon site. Felsenstein and Churchill *(44)* also proposed a model where the rates of substitutions at adjacent sites are correlated. Also, several models describe the evolution of pairs of interacting nucleotides among ribosomal RNA molecules *(45–47)*
2. It is not very clear which of the approach is the best for phylogenetic inference: non-deterministic or deterministic. Note however that deterministic methods are generally faster than stochastic optimisation approaches and sufficient for numerous optimisation problems *(48)*. On the other hand, stochastic methods have the ability to find several near-optimal trees, which gives an idea of the inferred tree variability.
3. Under the default settings, PhyML modifies the tree topology using NNI moves and simultaneously optimises branch lengths. However, when the tree topology estimate is stable (i.e. no improvement of the likelihood can be found by modifying the current tree topology), the optimisation concentrates on branch lengths and parameters of the Markov model in order to save computing time. NNI moves with optimisation of the central and the four adjacent branch lengths are also systematically tested during the very last optimisation step. This last step frequently finds a modification of the tree topology that was not detected by the other approximate (but fast) tree topology search methods (i.e. NNI, NNI+SPR or full-SPR).
4. The SPR-search strategy implemented in PhyML 3.0 actually relies on filtering SPR moves using the parsimony criterion instead of a distance-based approach. Indeed, parsimony and likelihood are closely related from a statistical perspective and

the analysis of real and simulated data (Dufayard, Guindon, Gascuel, unpublished) have demonstrated the benefits of using a parsimony-based filter.

5. PhyML binary file must be located in a directory listed in the PATH environment variable if the program is launched from a command-line window. The program can also be launched by typing "./phyml" provided that PhyML binary file is in the current directory. Launching PhyML by clicking on the corresponding icon is not recommended. In case PhyML can not find the sequence data file when launched by clicking on the icon, we suggest using a command-line window.
6. Questions regarding the amount of memory required can be eluded using the "-DBATCH" flag when compiling the program. This option is available through a simple modification of the Makefile. It is highly recommended to use this option when launching PhyML in batch mode or when comparing run times of different programs.
7. The estimation of bases or amino acid frequencies relies on a iterative method that takes into account gaps and ambiguous characters. The frequencies of the non-ambiguous characters (bases or amino acids) at step n are functions of the counts of the non ambiguous characters plus the counts of the ambiguous characters weighted by the probabilities of the non-ambiguous characters estimated at step $n-1$. These probabilities correspond to the frequencies estimated at step $n-1$. The same approach is also used in PAML *(49)* and PHYLIP *(12)* programs.
8. PhyML uses an original method that simultaneously estimates the gamma-shape parameter and the proportion of invariants (Guindon and Gascuel, unpublished). This method relies on the observation that the two parameters show a strong quasi-linear positive relationship. Basically, the gamma-shape parameter is first estimated using a standard one-dimensional optimisation method. The proportion of invariants is then deduced from the linear relationship with the gamma-shape parameter. Hence, the estimation of the proportion of invariants does not rely on time-consuming optimisation methods.

Acknowledgements

This work was supported by the "MITOSYS" grant from ANR. The chapter itself is the contribution 2007–08 of the Institut des Sciences de l'Evolution (UMR5554-CNRS).

References

1. Felsenstein, J. (1981) Evolutionary trees from DNA sequences: a maximum likelihood approach. *J Mol Evol* **17**, 368–76.
2. Rogers, J., and Swofford, D. (1999) Multiple local maxima for likelihoods of phylogenetic trees: a simulation study. *Mol Biol Evol* **16**, 1079–85.
3. Huelsenbeck, J. P., and Hillis, D. (1993) Success of phylogenetic methods in the four-taxon case. *Syst Biol* **42**, 247–64.
4. Swofford, D., Olsen, G., Waddel, P., and Hillis, D. (1996) Phylogenetic inference. In D. Hillis, C. Moritz, B. Mable, eds., *Molecular Systematics*, chapter 11. Sinauer, Sunderland, MA.
5. Guindon, S., and Gascuel, O. (2003) A simple, fast and accurate algorithm to estimate large phylogenies by maximum likelihood. *Syst Biol* **52**, 696–704.
6. Olsen, G., Matsuda, H., Hagstrom, R ., and Overbeek, R. (1994) fastDNAml: a tool for construction of phylogenetic trees of DNA sequences using maximum likelihood. *Comput Appl Biosci* **10**, 41–8.
7. Hordijk, W., and Gascuel, O. (2005) Improving the efficiency of SPR moves in phylogenetic tree search methods based on maximum likelihood. *Bioinformatics* **21**, 4338–47.
8. Anisimova, M., and Gascuel, O. (2006) Approximate likelihood-ratio test for branches: a fast, accurate, and powerful alternative. *Syst Biol* **55**, 539–52.
9. Shimodaira, H., and Hasegawa, M. (1999) Multiple comparisons of log-likelihoods with applications to phylogenetic inference. *Mol Biol Evol*, **16**, 1114–6.
10. Jukes, T., and Cantor, C. (1969) Evolution of protein molecules. In H. Munro, ed., *Mammalian Protein Metabolism*, volume III, chapter 24, 21–132. Academic Press, New York.
11. Kimura, M. (1980) A simple method for estimating evolutionary rates of base substitutions through comparative studies of nucleotide sequences. *J Mol Evol* **16**, 111–20.
12. Felsenstein, J. (1993) *PHYLIP (PHYLogeny Inference Package) Version 3.6a2.* Distributed by the author, Department of Genetics, University of Washington, Seattle.
13. Hasegawa, M., Kishino, H., and Yano, T. (1985) Dating of the Human-Ape splitting by a molecular clock of mitochondrial-DNA. *J Mol Evol* **22**, 160–74.
14. Tamura, K., and Nei, M. (1993) Estimation of the number of nucleotide substitutions in the control region of mitochondrial DNA in humans and chimpanzees. *Mol Biol Evol* **10**, 512–26.
15. Lanave, C., Preparata, G., Saccone, C., and Serio, G. (1984) A new method for calculating evolutionary substitution rates. *J Mol Evol* **20**, 86–93.
16. Tavaré, S. (1986) Some probabilistic and statistical problems on the analysis of DNA sequences. *Lect Mathe Life Sci*, **17**, 57–86.
17. Whelan, S., and Goldman, N. (2001) A general empirical model of protein evolution derived from multiple protein families using a maximum-likelihood approach. *Mol Biol Evol* **18**, 691–9.
18. Dayhoff, M., Schwartz, R., and Orcutt, B. (1978) A model of evolutionary change in proteins. In M. Dayhoff, ed., *Atlas of Protein Sequence and Structure*, volume 5, 345–52. National Biomedical Research Foundation, Washington, D. C.
19. Jones, D., Taylor, W., and Thornton, J. (1992) The rapid generation of mutation data matrices from protein sequences. *Comput Appl Biosci*, **8**, 275–82.
20. Henikoff, S., and Henikoff, J. (1992) Amino acid substitution matrices from protein blocks. *Proc Natl Acad Sci USA* **89**, 10915–9.
21. Adachi, J., and Hasegawa, M. (1996) MOLPHY version 2.3. programs for molecular phylogenetics based on maximum likelihood. In M. Ishiguro, G. Kitagawa, Y. Ogata, H. Takagi, Y. Tamura, T. Tsuchiya, eds., *Computer Science Monographs*, 28. The Institute of Statistical Mathematics, Tokyo.
22. Dimmic, M., Rest, J., Mindell, D., and Goldstein, D. (2002) rtREV: an amino acid substitution matrix for inference of retrovirus and reverse transcriptase phylogeny. *J Mol Evol* **55**, 65–73.
23. Adachi, J., P., Martin, W., and Hasegawa, M. (2000) Plastid genome phylogeny and a model of amino acid substitution for proteins encoded by chloroplast DNA. *J Mol Evol* **50**, 348–58.
24. Kosiol, C., and Goldman, N. (2004) Different versions of the Dayhoff rate matrix. *Mol Biol and Evol* **22**, 193–9.
25. Muller, T., and Vingron, M. (2000) Modeling amino acid replacement. *J Comput Biol* **7**, 761–76.

26. Cao, Y., Janke, A., Waddell, P., Westerman, M., Takenaka, O., Murata, S., Okada, N., Paabo, S., and Hasegawa, M. (1998) Conflict among individual mitochondrial proteins in resolving the phylogeny of eutherian orders. *J Mol Evol* **47**, 307–22.
27. Yang, Z. (1994) Maximum likelihood phylogenetic estimation from DNA sequences with variable rates over sites: approximate methods. *J Mol Evol* **39**, 306–14.
28. Gascuel, O. (1997) BIONJ: an improved version of the NJ algorithm based on a simple model of sequence data. *Mol Biol Evol* **14**, 685–95.
29. Posada, D., and Crandall, K. (1998) Modeltest: testing the model of DNA substitution. *Bioinformatics* **14**, 817–918.
30. Abascal, F., Zardoya, R., and Posada, D. (2005) Prottest: selection of best-fit models of protein evolution. *Bioinformatics* **21**, 2104–5.
31. Galtier, N., and Jean-Marie, A. (2004) Markov-modulated Markov chains and the covarion process of molecular evolution. *J Comput Biol*, **11**, 727–33.
32. Lin, Y.-H., McLenachan, P., Gore, A., Phillips, M., Ota, R., Hendy, M., and Penny, D. (2002) Four new mitochondrial genomes, and the stability of evolutionary trees of mammals. *Mol Biol Evol* **19**, 2060–70.
33. Reyes, A., Gissi, C., Catzeflis, F., Nevo, E., Pesole, G., and Saccone, C. (2004) Congruent mammalian trees from mitochondrial and nuclear genes using bayesian methods. *Mol Biol Evol* **21**, 397–403.
34. Murphy, M., Eizirik, E., O'Brien, S., Madsen, O., Scally, M., Douady, C., Teeling, E., Ryder, O., Stanhope, M., de Jong, W., and Springer, M. (2001) Resolution of the early placental mammal radiation using bayesian phylogenetics. *Science* **294**, 2348–51.
35. Delsuc, F., Scally, M., Madsen, O., Stanhope, M., de Jong, W., Catzeflis, F., Springer, M., and Douzery, E. (2002) Molecular phylogeny of living xenarthrans and the impact of character and taxon sampling on the placental tree rooting. *Mol Biol Evol* **19**, 1656–71.
36. Amrine-Madsen, H., Koepfli, K., Wayne, R., and Springer, M. (2003) A new phylogenetic marker, apolipoprotein B, provides compelling evidence for eutherian relationships. *Mol Phylogenet Evol* **28**, 225–40.
37. Springer, M., Bry, R. D., Douady, C., Amrine, H., Madsen, O., de Jong, W., and Stanhope., M. (2001) Mitochondrial versus nuclear gene sequences in deep-level mammalian phylogeny reconstruction. *Mol Biol Evol* **18**, 132–43.
38. D'Erchia, A., Gissi, C., Pesole, G., Saccone, C., and Arnason, U. (1996) The guinea-pig is not a rodent. *Nature* **381**, 597–600.
39. Reyes, A., Pesole, G., and Saccone, C. (1998) Complete mitochondrial DNA sequence of the fat dormouse, Glis glis: further evidence of rodent paraphyly. *Mol Biol Evol* **15**, 499–505.
40. Reyes, A., Pesole, G., and Saccone, C. (2000) Long-branch attraction phenomenon and the impact of among-site rate variation on rodent phylogeny. *Gene* **259**, 177–87.
41. Philippe, H. (1997) Rodent monophyly: pitfalls of molecular phylogenies. *J Mol Evol* **45**, 712–5.
42. Sullivan, J., and Swofford, D. (1997) Are guinea pigs rodents? The importance of adequate models in molecular phylogenetics. *J Mammal Evol* **4**, 77–86.
43. Felsenstein, J. (1985) Confidence limits on phylogenies: an approach using the bootstrap. *Evolution* **39**, 783–91.
44. Felsenstein, J., and Churchill, G. (1996) A hidden Markov model approach to variation among sites in rate of evolution. *Mol Biol Evol* **13**, 93–104.
45. Schniger, M., and von Haesler, A. (1994) A stochastic model for the evolution of autocorrelated DNA sequences. *Mol Phylogeny Evol* **3**, 240–7.
46. Muse, S. (1995) Evolutionary analyses of DNA sequences subject to constraints on secondary structure. *Genetics* **139**, 1429–39.
47. Tillier, E., and Collins, R. (1998) High apparent rate of simultaneous compensatory base-pair substitutions in ribosomal rna. *Genetics* **148**, 1993–2002.
48. Aarts, E., and Lenstra, J. K. (1997) *Local Search in Combinatorial Optimization*. Wiley, Chichester.
49. Yang, Z. (1997) PAML : a program package for phylogenetic analysis by maximum likelihood. *Comput Appl Biosci* **13**, 555–6.

Chapter 7

Trees from Trees: Construction of Phylogenetic Supertrees Using Clann

Christopher J. Creevey and James O. McInerney

Abstract

Supertree methods combine multiple phylogenetic trees to produce the overall best "supertree." They can be used to combine phylogenetic information from datasets only partially overlapping and from disparate sources (like molecular and morphological data), or to break down problems thought to be computationally intractable. Some of the longest standing phylogenetic conundrums are now being brought to light using supertree approaches. We describe the most widely used supertree methods implemented in the software program "clann" and provide a step by step tutorial for investigating phylogenetic information and reconstructing the best supertree. Clann is freely available for Windows, Mac and Unix/Linux operating systems under the GNU public licence at http://bioinf.nuim.ie/software/clann.

Key words: Supertree software, phylogenetic reconstruction, phylogeny, congruency test, phylogenetic signal detection.

1. Introduction

1.1. What Are Supertrees?

Supertree methods combine the information from a set of taxonomically overlapping phylogenetic trees sometimes called source, or input trees and produce a supertree, or set of equally good supertrees, containing a complete set of all leaves found in the input trees. The analysis requires that the source trees be connected by sets of shared taxa. Source trees that share no taxa in common cannot be combined; however, two non-overlapping source trees may be "bridged" by a third that shares taxa with both (**Fig. 7.1**).

In terms of phylogenetic methods, supertrees are in their infancy, the first papers outlining methods only appeared in the

David Posada (ed.), *Bioinformatics for DNA Sequence Analysis,* Methods in Molecular Biology 537

DOI 10.1007/978-1-59745-251-9_7

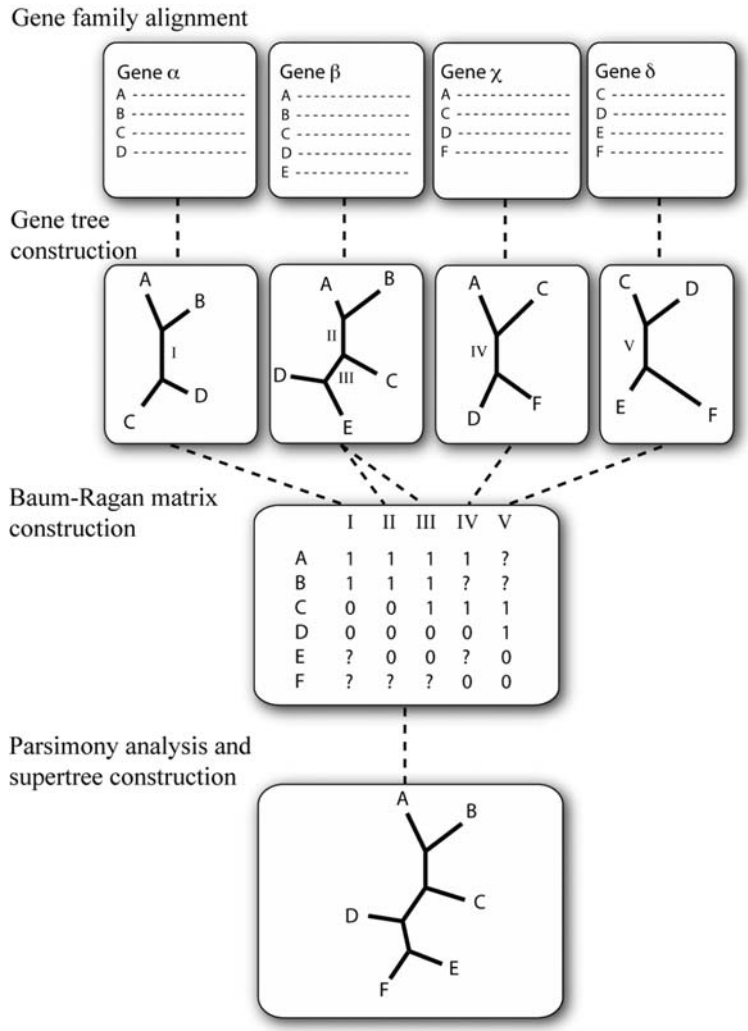


Fig. 7.1. Matrix representation with parsimony (MRP) method.The MRP procedure is as follows: once the alignment of the gene families is complete, trees are built for each of the genes separately. Within each of these trees, the internal branches (or splits) are identified (I–V above). A Baum-Ragan coding scheme is constructed, containing a column for each of the internal branches. The coding scheme groups the taxa into those that appear on either side of the split. For instance for internal branch I, taxa A and B are on one side and taxa C and D are on the other. In the coding scheme, taxa A and B are both marked with a "1" and taxa C and D are both marked with a "0." As taxa E and F are not in this tree, they are marked with a "?" in column I. When the matrix is completed, a maximum parsimony approach is generally used to reconstruct the supertree.

early 1980s *(1, 2)*. They are generally characterised by a set of rules detailing how the phylogenetic information from the source trees may be combined. Different methods use different rules but the end result should allow not only the combination of information contained in the source trees, but also the inference of

relationships not present in any one source tree. It is also desirable that the resulting supertree does not contain any relationship contradicted in every source tree *(3)*.

Supertree methods can be considered a generalisation of consensus methods, except they combine information from partially overlapping trees. However, given a set of completely overlapping source trees (the so-called "consensus setting"), a supertree method should work exactly like a consensus method in combining the phylogenetic information.

1.2. Why Would You Want to Make Supertrees?

1.2.1. Cumulative Evidence Is Philosophically Preferable

There are both philosophical and practical reasons for preferring to use a supertree method rather than other alternatives. From a philosophical viewpoint, the more data you can use to solve a problem, the better the result is likely to be. Supertree methods allow the inclusion of information from disparate sources and this opens the possibility for relationships to be inferred – a situation that would not be possible from one data source alone. For instance, supertree methods have been used to combine genetic data with morphological data *(4, 5)*. This combination results in datasets that contain both macro-evolutionary and micro-evolutionary information allowing statements to be made about relationships over evolutionary distances not possible with either of the datasets alone.

From a practical viewpoint, few datasets contain exactly the same species/strains/proteins and so their combination can be difficult. Furthermore, patchy gene distribution means that only a few (estimated at 1% *(6)*) genes are universally distributed in single copy. Traditionally phylogenetic studies have relied on this 1% to reconstruct phylogenies of organisms. The ideal situation would be to reconstruct a phylogenetic tree using 100% of the available data. Supertrees provide the only realistic possibility of using 100% of information to reconstruct the tree of life, as a concatenated alignment of all genes from all organisms is likely to have too much missing data. The only restriction to using supertree methods is that it must be possible to represent the data as a tree.

1.2.2. To Identify Trees That Are Similar and Trees That Are Different

In real biological datasets, the biggest problem is not data overlap but the conflict that exists between different data types. Conflicting phylogenetic signals can occur because of model misspecification in generating the source trees, hidden paralogy, poor homology determination, lineage sorting or horizontal gene transfer (HGT) *(7–9)*, causing gene trees to differ from the "true" history of speciation events (the species tree). If the factors causing the differences in the gene trees are randomly distributed, then the combination of many gene trees should reveal the species tree. However, when the differences are too great it is not always desirable to simply combine the information; a more investigative

approach is desirable in order to identify level of compatibility within the individual source trees. The classic example of this problem is HGT in genomic information. A lot has been said recently about the role of HGT in the evolution of microorganisms *(7, 10–16)*. Opinions differ over the role HGT has played and whether it has obliterated any possibility of accurately reconstructing the tree of life using the majority of genes *(17, 18)*.

It is possible to investigate the role of HGT in a dataset using supertree methods *(7)*. Using this approach, trees made from orthologous genes are used to identify the overall phylogenetic signal existing in the majority of genes. Next, the gene trees that differ significantly from this signal can be identified. At this point, depending on the goal of our research, we could examine the genes individually to see why they differ, or see if there is an alternative phylogenetic signal underlying the first. An example of this approach was the recent work on the origin of Eukaryotes, where individual signals were stripped from the data and the secondary and subsequent phylogenetic signals were examined *(19)*.

The same approach works for datasets affected by "hidden paralogy," a situation where deletions of paralogs on different lineages can result in the remaining paralogs being misclassified as orthologs. Long branch attraction *(20)*, systematic biases caused by the GC content of synonymous sites *(21)* or evolutionary model misspecification *(22)* also result in spurious source tree topologies and can be identified by investigating the data further.

1.2.3. Divide and Conquer? Lots of Small Trees Are Easier to Construct than One Big Tree

Biological datasets are becoming progressively larger and more computationally difficult to handle. A divide-and-conquer approach therefore is sometimes necessary whereby a single large problem is divided into smaller and easier to handle subsets. Each of these subsets can then be solved independently and the results combined to form the overall solution. Supertree methods are well suited to this approach. Lots of small trees are easier to construct than one big tree and their combination into a single tree is exactly the purpose for which supertree methods were designed.

1.3. Are There Alternatives?

Other methods exist that can tackle the same type of problems as supertree methods.

1.3.1. Supermatrix

Concatenating alignments together to produce a supermatrix is a very popular approach to combine information from different sources *(23, 24)*. These methods work best when the alignments to be combined have very little or no missing data. The appeal of this approach is that the resulting tree is created directly from the sequences without the necessity for any intermediate step such as the construction of source trees as happens in a supertree analysis. Another alluring feature of the supermatrix approach is based on the assumption that misleading evidence of phylogeny

(homoplasy) will be random, whereas true phylogenetic signal, however weak, will be additive and with enough data this signal should emerge.

Weaknesses of this approach include the inability to explicitly deal with missing data leading to a situation where we don't know the effect that differing levels of missing data have on the result, although some have attempted to estimate this affect *(25)*. Finally, the analysis of a supermatrix requires much more computational power than the divide-and-conquer approach taken by supertree methods. Because of these computational requirements, it is sometimes not possible to build a tree from a concatenated alignment.

1.3.2. Genome Content

There are several other methods available for reconstructing a tree when using data from whole genomes. The most commonly used method is called a genome content approach *(26)*. In this analysis, gene families are identified in the genomes of interest and the presence (and sometimes number of copies) of genes from each family in each genome is recorded. Genomes from closely related species are expected to have similar genome content. This information is combined into a data-matrix that can be used to build a phylogeny. Another variant of this approach is a gene order analysis *(27)*. Here, the order rather than the presence or absence of genes is used as a phylogenetic marker. This is a more "fine-grained" approach than genome content analyses and is better suited to reconstructing relationships between closely related organisms.

The advantage of genome content methods is that they are computationally more tractable than supermatrix or supertree approaches and they have the ability to use all the information from all the genes in a genome. The disadvantages include being very "coarse-grained" and not using the phylogenetic information from the genes themselves, and being very sensitive to hidden paralogy including HGT. Furthermore, even though there are a few simple but interesting models of whole genome evolution *(28–30)*, we have no realistic models at this scale *(8)* and so we must rely on basic parsimony principles the majority of the time to reconstruct the overall tree.

1.3.3. Conditioned Reconstruction

Recently a method of using whole genomes to reconstruct organismal relationships called "conditioned reconstruction" has been proposed *(31)*. Similar to genome content methods, conditioned reconstruction uses information of presence or absence of genes, but between pairs of genomes. This information is further enriched with rates of gene loss and gain within the frequencies of presence or absence. A "conditioning" genome is used to calculate these frequencies between it and all other genomes in the analysis. To calculate the shared absence of genes between

genomes several conditioning genomes are used and the combined information is used to reconstruct phylogenetic relationships *(8)*. The advantages to this approach are an increased sensitivity to more recent relationships over standard genome content methods while still retaining the ability to use all the information within all the genomes. Disadvantages include our lack of knowledge of how well it performs under violations to the assumptions made by the method. Also it does not use the information that may be gained by directly comparing the sequences of the genes used *(8)*.

1.4. What Supertree Methods Are Out There?

A variety of supertree methods have been developed since the original publications. Each method approaches the problem of combining the information from multiple trees in different ways.

1.4.1. Matrix Representation

The most widely used supertree method is based upon a method proposed independently by both Baum *(32)* and Ragan *(33)*. Called matrix representation with parsimony (MRP), it uses a coding scheme to construct a matrix representing the relationships within the source trees. Typically, a maximum parsimony algorithm is then used to reconstruct a supertree from this matrix.

The method identifies the internal branches (also called splits) within each of the source trees and a simple coding scheme of 1s and 0s are used to determine which taxa are on either side of the split (**Fig. 7.1**). All the taxa on one side of the split are marked with a 1 and the taxa on the other side of the split are marked with a 0. Any taxa not present on this source tree are marked with a "?" (**Fig. 7.1**). For unrooted source trees (as is most common with genomic data) it makes no difference which side of a split is marked with a 1 or a 0. The coding for all the internal branches across all the source trees are then combined into a single matrix and this is used to reconstruct the supertree. Despite being widely used, there have been major criticisms about biases in this method, including a tendency to favour the relationships of larger source trees than smaller trees and towards source trees with certain phylogenetic shapes *(3)*.

1.4.2. Average Consensus

A second approach to reconstructing a supertree involves calculating distance matrices representing the relationships within the source trees. These methods may make use of the branch lengths on the source trees and can result in a supertree with branch lengths. One such method is called "average consensus" *(34)*. In this approach the path-length distances of each taxon to every other taxon is calculated across each of the source trees. The average distance of each taxon to every other is then used in a final distance matrix, from which a supertree is constructed (**Fig. 7.2**). Sometimes however there

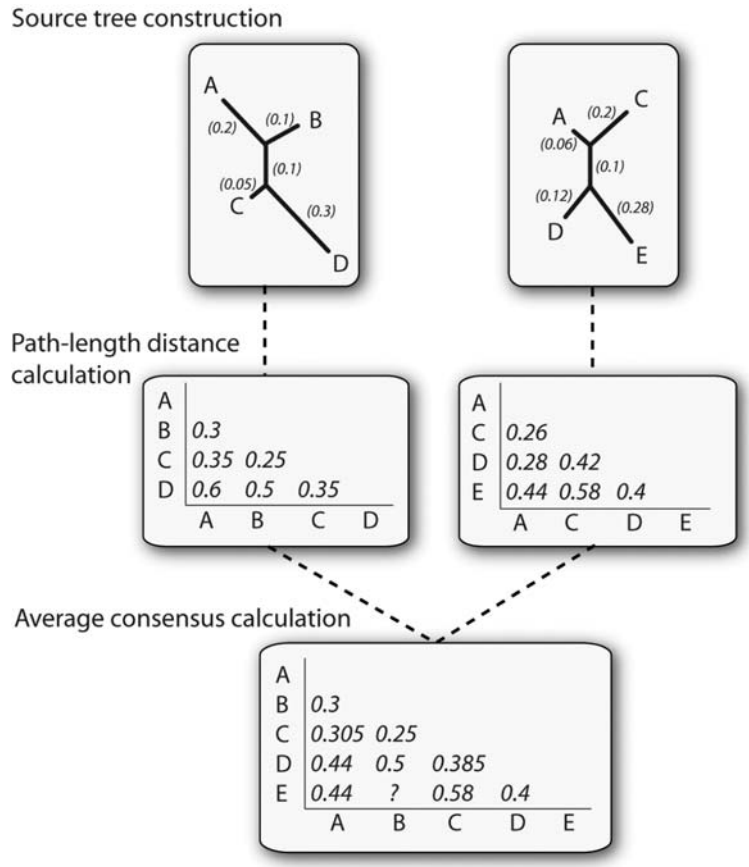


Fig. 7.2. Calculation of average consensus. In this approach, the branch lengths from the source trees are used to calculate the path-length distances of each taxon to every other taxa. In the trees above, the numbers in brackets indicate the lengths of their associated branches. For example, the source tree on the left has a path-length distance from taxon A to taxon D of 0.6 (0.2 + 0.1 + 0.3). The average distance from each taxon to every other taxon is then calculated for the average consensus. For example, the distance from taxon A to taxon C in the two source trees are 0.35 and 0.26 respectively. The average of these (0.305) is the result put in the average consensus matrix. Some distances are not possible to calculate because the taxa do not appear together in any tree (like with Taxa B and E above); in these cases the value is estimated from the surrounding values in the average consensus matrix.

is an example where two taxa never appear together on any source tree, in this case the average distance of each of these taxa to taxa they both share in common is used to estimate the distance that they would be from each other, if they appeared together on a tree. This is essentially "filling in the blanks" where we have no information concerning their evolutionary relationships to each other. Several methods have been developed to calculate these missing values, *see (35)* for

more details. Once the average consensus matrix is complete, a variety of distance-based phylogenetic methods can be used to reconstruct the best supertree. The most commonly used is a least-squares fit *(35)*, but it is also possible to use a simple neighbour-joining (NJ) algorithm *(36)*. The advantage of this method is that it produces a supertree with branch lengths.

1.4.3. MSSA-Type

Other distance-matrix based methods use different approaches to find the best supertree. The most similar supertree algorithm (MSSA) *(7)* searches for the best supertree without averaging the information from the source trees. Instead, a heuristic search of supertree-space is carried using a scoring function, which when minimised, returns the supertree that is the most similar to the set of source trees. This scoring function works by comparing a candidate supertree to each of the source trees individually. As the supertree will contain all taxa and any source tree is likely to contain only a subset, for each comparison to a source tree the candidate supertree is pruned down to the same taxon set as the source tree. A direct comparison is then possible between the pruned candidate supertree and the source tree. The difference between the two trees is calculated by summing the absolute differences between the path-length distance matrices of the two trees. In this case the path length is defined by the number of internal nodes separating any two taxa on a tree (**Fig. 7.3**). This pruning-then-comparison method is carried out against every source tree and the sum of the absolute differences is used as a score representing the similarity of the candidate supertree to the set of source trees. A score of zero represents the situation where every source tree is identical to the supertree (when the supertree is pruned to the same size for comparison). Multiple candidate supertrees are tested to find the one that minimises the score function when compared to the source trees. An exhaustive search of supertree-space can be carried out or a standard heuristic search can be used (for instance: nearest neighbour interchange (NNI) or sub-tree pruning and regrafting (SPR)).

1.4.4. Quartets

Quartet methods generally break down the source trees into their constituent quartets and use various approaches to find the supertree that shares the most quartets with the set of source tree quartets. A set of quartets is all possible 4-taxon trees that can be made by pruning the tree. The optimum supertree can be found using several techniques, including simply counting the number of shared quartets or by using a "puzzling step" whereby random subsets of quartets are combined in a step-wise manner to "grow" the supertree. This is then repeated many times to see which supertree relationships are reconstructed most often.

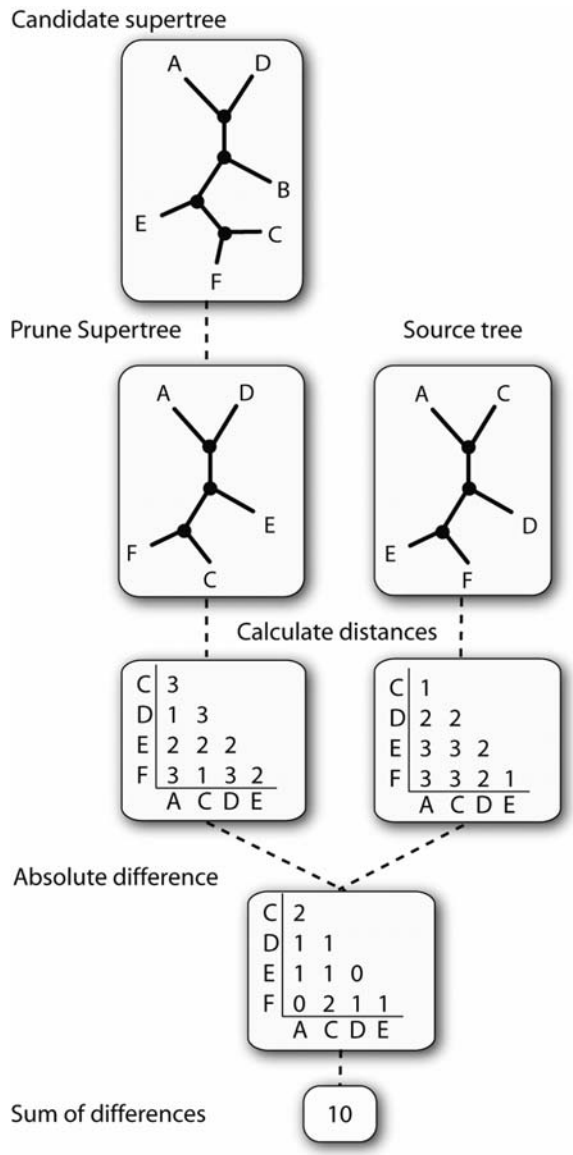


Fig. 7.3. Most Similar Supertree Algorithm (MSSA). In this approach, a function is used to assess candidate supertrees. A heuristic or exhaustive search of supertree space is carried out and the supertree that minimises the function is the most similar supertree to the set of source trees. The difference between the candidate supertree and each source tree is calculated separately and the sum of these scores is the overall score for the supertree. For each comparison to a source tree the supertree is first pruned down to the same taxa set as the source tree (above). Next a path-length distance score representing the differences between the two trees is calculated. The path-length distances are the number of internal nodes (*filled circles* in the trees above) that are in the path between any two taxa on the tree. The sum of the absolute differences between the matrices is the score representing the difference between the supertree and this source tree. This value is usually divided by either the number of comparisons in the matrix or the number of species shared by the supertree and the source tree to counteract biases from large source trees.

2. Program Usage

Clann *(37)* is a command-line software package for investigating phylogenetic information through supertree analyses and is freely available under a GNU public license agreement. Version 3.1 implements five different supertree methods, including MRP, average consensus and the MSSA. In this version of clann the MRP criterion requires the use of an external parsimony program like PAUP* *(38)*; future versions will remove this requirement.

2.1. Installation

Clann is available at http://bioinf.nuim.ie/software/clann. On the download page the choice of three different operating systems are available (Mac OSX, Linux and Microsoft Windows). In Mac OSX an installation script is included, which installs the readline and ncurses libraries (if needed) before putting clann into the folder /usr/bin/. Once in this location (and the user starts a new terminal window) clann will be visible to the operating system from any directory. An administrative password will be needed to successfully install clann.

On a Linux operating system, the clann program should either be located in the same directory as the input files, or somewhere on your path (e.g. *~/bin/* or */usr/local/bin*). If you do not know which directories are on your path, ask your system administrator.

On the Microsoft Windows double clicking on the icon associated with clann will run the program. Using this operating system, the clann program *must* be located in the same directory as the input files, an alias or shortcut to clann will not suffice.

To run clann on the MacOSX or Linux operating systems, type the command "*./clann*" or "*clann*" in a terminal window.

2.2. File Formats

Clann accepts source trees in two different formats: newick (also called phylip format) and nexus format (*see* **Note 1**). Multiple trees can be contained in the same file in both formats. Newick-formatted trees are the simplest to construct and can contain branch lengths, internal branch labels, tree weights and tree names (*see* **Note 2**) (**Fig. 7.4**).

The nexus format is a modular system for representing many types of systematic information, including sequences and trees *(39)*. Branch lengths and internal branch labels are indicated in the same manner to newick-formatted files. Different types of systematic information are contained within "blocks." It is also possible to include a clann block, containing the commands to be executed on the data in the file. To load a file of trees into clann use the command exe filename or include the name of the file to be executed along with the call for clann at the operating system

i) (A,B,(C,D));
ii) (A:0.01,B:0.02,(C:0.01,D:0.03):0.01);
iii) (A,B,(C,D)Int1);
iv) (A,B,(C,D))[1.5];
v) (A,B,(C,D)); [Tree name]
vi) (A:0.01,B:0.02,(C:0.01,D:0.03)Int1:0.01)[1.5]; [Tree name]

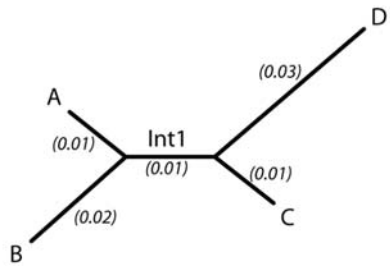


Fig. 7.4. Newick (phylip) format trees. Newick-formatted trees can contain a variety of information. (i) The simplest form which just contains the tree topology. (ii) Branch length information incorporated onto the tree (in brackets on the tree above). (iii) Internal branch name (Int1) included; this could also be used to indicate a bootstrap proportion value. (iv) A tree weight can be included within square brackets before the semicolon. This may be used if more or less emphasis should be applied to the relationships any tree (1 is the default). (v) Trees may be given specific names within square brackets after the semicolon, representing the datasets from which they were constructed. (vi) All possible information from (i) to (v) above included on a single tree. Multiple trees can be contained in a single file.

prompt (i.e. "clann filename") (*see* **Note 3**). After completing a summary of the relationships between the trees in the file, clann will return the prompt "clann>." From this point all the different commands available can be executed (*see* **Notes 4** and **5**).

2.3. Building Supertrees

Each supertree method in clann is a separate *criterion* on which operations such as heuristic searches, exhaustive searches or bootstrap resampling analyses can be carried out (*see* **Note 6**). The five criteria implemented in version 3.1 of clann are MRP (matrix representation with parsimony), DFIT (Most similar supertree algorithm), SFIT (split fit algorithm), QFIT (quartet fit algorithm) and AVCON (average consensus method).

2.3.1. Constructing an MRP Tree

From the clann prompt the command set criterion=mrp tells clann that all following commands are to use MRP as the criterion for assessing supertrees (*see* **Note 7**). The quickest way to reconstruct a supertree in this criterion is using a heuristic search of tree-space. The command hs ? lists the possible options for a heuristic search (**Fig. 7.5**).

```
hs (or hsearch) [options]

        Options         Settings                        Current
        ================================================================

        analysis        parsimony | nj                  *parsimony

        Parsimony options:
        weighted        yes | no                        *no
        swap            nni | spr | tbr                 *tbr
        addseq          simple | closest | asis |
                        random | furthest               *random
        nreps           <integer number>                *10

        General Options:
        savetrees       <filename>                      MRP.tree

                                                        *Option is nonpersistent
        ================================================================
```

Fig. 7.5. Options available with the heuristic search (hs) command using the matrix representation with parsimony (MRP) criterion.

By default a heuristic search will use PAUP* *(38)* to carry out the parsimony analysis. Clann will try to run PAUP* but if it fails to do so, it will return an error and suggest that the user should execute the created Baum-Ragan matrix in PAUP* separately. The use of other parsimony programs is also possible. The best supertree(s) is saved to the file "MRP.tree," although it is possible to change the name of the file using the option

```
savetrees=new-file-name.
```

2.3.2. Making an Average Consensus Tree

The quickest method to construct a supertree is to create a NJ tree using the average consensus method to create the distance matrix. This can be carried out under any criteria using the command nj. The resulting tree is both displayed on screen and saved to the file "NJtree.ph." The options for this approach only concern the methods used to fill in the missing values in the distance matrix. Typing the command nj ? returns the options possible with this command (**Fig. 7.6**).

```
nj [options]

        Options         Settings                        Current
        ================================================================

        missing         4point | ultrametric            *4point
        savetrees       <file name>                     *NJtree.ph

                                                        *Option is nonpersistent
        ================================================================
```

Fig. 7.6. Options available with the neighbour-joining supertree (nj) option.

To carry out a full average consensus analysis it is necessary to change the criterion to "avcon." By default, clann uses PAUP* *(38)* to carry out the heuristic search using the least-squared objective function. The resulting tree is displayed on screen and saved to the file "Heuristic_result.txt."

2.3.3. Making an MSSA Tree

The MSSA algorithm is called under the criterion "DFIT" in clann. The command hs ? displays the options available under the MSSA criterion (*see* **Note 8**) (**Fig. 7.7**).

By default clann will create ten NJ trees with some random changes and carry out the heuristic search from these starting points. It is also possible to specify a random pre-sample of supertree space to find the best starting points from which to carry out the heuristic search using the option "start." When the search is complete, the best supertree(s) are displayed on screen.

2.4. Visualising Output

By default, any supertree reconstructed by clann is saved into its respective files in newick format. There are a variety of tree-viewing applications that can read these files including stand-alone applications like Treeview *(40)* and online tools like iTOL *(41)*. Clann also saves the trees returned from heuristic searches as a post-script image file (called "trees.ps") that can be viewed by a variety of applications.

2.5. Interrogating Input Trees

One of the strengths of clann is its ability to allow the user to investigate the phylogenetic support in the source trees for a supertree (*see* **Notes 9 and 10**). At the most basic level, the user can choose to rank the source trees according to their similarity to the best supertree. This is carried out during a heuristic search for the best supertree. If the option "drawhistogram" has been set to "yes," a histogram providing information on how similar the

```
hs (or hsearch) [options]

        Options         Settings                        Current
        ================================================

        sample          <integer number>                *10,000
        nreps           <integer number>                *10
        swap            nni | spr                       spr
        nsteps          <integer number>                3
        start           nj | random | <filename>        nj
        maxswaps        <integer number>                *1,000,000
        savetrees       <filename>                      Heuristic_result.txt
        weight          equal | comparisons             comparisons
        drawhistogram   yes | no                        *no
        nbins           <integer number>                *20
        histogramfile   <filename>                      *Heuristic_histogram.txt

                                                        *Option is nonpersistent
        ================================================
```

Fig. 7.7. Options available with the heuristic search (hs) command using the most similar supertree algorithm (MSSA) criterion.

```
0.00 - 0.06     |================================== (623)
0.07 - 0.14     | (0)
0.15 - 0.21     | (0)
0.22 - 0.28     | (0)
0.29 - 0.36     | (2)
0.37 - 0.43     | (11)
0.44 - 0.50     | (11)
0.51 - 0.58     |== (45)
0.59 - 0.65     |= (30)
0.66 - 0.72     |= (25)
0.73 - 0.80     | (8)
0.81 - 0.87     | (4)
0.88 - 0.94     | (2)
0.95 - 1.02     | (4)
1.03 - 1.09     | (5)
1.10 - 1.16     | (4)
1.17 - 1.23     | (3)
1.24 - 1.31     | (0)
1.32 - 1.38     | (1)
1.39 - 1.45     | (2)
```

Fig. 7.8. Histogram detailing the similarity of the best supertree to the source trees.

supertree is to the set of source trees is displayed (where a score of 0 means they are identical) (**Fig. 7.8**). The information is also saved to the file "Heuristic_histogram.txt."

A bootstrap analysis of the source trees can also be carried out in clann using the command bootstrap or boot. This analysis will resample the set of source trees with replacement to create a new set with the same number of trees as the original. This is generally carried out 100 times and the best supertree is found for each. Clann then carries out a summary of the source trees (usually a majority rule consensus) and the relationships with the best support are displayed to screen. All the best supertrees for each bootstrap replicate are saved to the file "bootstrap.txt" and the consensus to "consensus.txt." The bootstrap analysis can be carried out for each of the five different criteria in clann. Typing boot? displays the options available under the current criterion (**Fig. 7.9**).

Finally it is possible to carry out an analysis of the level of congruent phylogenetic signal across the set of source trees using a permutation-tail-probability test. The test implemented in clann is called the YAPTP (yet another permutation-tail-probability) test. This test compares the score of the best supertree to the score of the best supertrees from 100 randomly permuted versions of the source trees. For each of the 100 replicates of this analysis each source tree is randomised, thereby destroying the any congruent signal between them, while keeping the same taxon distribution and source tree sizes. A heuristic search for the best supertree for each of these randomised datasets is then carried out. If topological congruence between the source trees is better

```
bootstrap (or boot) [options]

        Options         Settings                          Current
        ================================================================

        nreps           <integer number>                  *100
        hsreps          <integer number>                  *10
        sample          <integer number>                  *10,000
        swap            nni | spr | all                   spr
        start           random | <filename>               random
        nsteps          <integer number>                  3
        treefile        <output treefile name>            bootstrap.txt
        maxswaps        <integer number>                  *1,000,000
        weight          equal | comparisons               comparisons
        consensus       strict | majrule | minor | <proportion> *majrule
        consensusfile   <filename>                        consensus.ph


                                                          *Option is nonpersistent
        ================================================================
```

Fig. 7.9. Options available with the bootstrapping (boot) command using the most similar supertree algorithm (MSSA) criterion.

than random, then the supertree score for the real dataset is expected to be better than any of the supertree scores from the randomised datasets. This is essentially testing that the phylogenetic signal shared between the source trees is better than random noise *(7)*.

3. Examples

The datasets used in the following examples are from *(7)*. This analysis concerns the extent of phylogenetic signal within the Prokaryotes. Two datasets were constructed, the first consisting of single-copy gene families from ten genomes within the Gammaproteobacteria and the second consisting of single-copy gene families from 11 genomes spanning the earliest branches of the prokaryotic tree of life. For more details on the methods used to create the source trees *see (7)*. The source trees created are available for download at http://bioinf.nuim.ie/supplementary/royalsoc04/.

Beginning with the dataset from genomes spanning the earliest branches of the Prokaryotes (the file named "11taxonfundamentals.ph.txt"), from within clann type:

```
exe 11taxonfundamentals.ph.txt
```

Clann will read in the source trees and calculate and display some basic statistics about the data (**Fig. 7.10**).

```
Reading Newhampshire (Phylip) format source tree file

        Source tree summmary:

        ---------------------------------------------------
                Number of input trees: 198
                Number of unique taxa: 11
                Total unrooted trees in Supertree space?
                        3.44594e+07

        Occurrence summary:

                number  Taxa name                       Occurrence

                0       A.aeolic                        141
                1       B.burgdo                        85
                2       S.aureus                        137
                3       Synechoc                        148
                4       M.tuberc                        144
                5       D.radiod                        155
                6       C.jejuni                        142
                7       EcoliK12                        179
                8       M.pulmon                        67
                9       C.pnL029                        93
                10      Halobact                        66


        Co-occurrence summary:

                Taxa Number

                     0    1    2    3    4    5    6    7    8    9    10
                0    -
                1    75   -
                2    101  71   -
                3    116  69   108  -
                4    104  66   107  120  -
                5    108  72   120  124  126  -
                6    121  77   102  116  102  112  -
                7    133  81   126  135  128  140  136  -
                8    55   51   64   56   59   60   55   63   -
                9    85   62   72   80   72   77   82   89   50   -
                10   48   27   51   53   53   52   43   58   26   31   -


        Source tree size summary:

  num leaves
        4   |================================== (52)
        5   |================== (28)
        6   |============== (20)
        7   |============ (18)
        8   |============ (18)
        9   |========== (16)
        10  |==================== (31)
        11  |========== (15)


        ---------------------------------------------------
  clann>
```

Fig. 7.10. Output generated by clann after the execution of a phylip-formatted file of multiple source trees.

3.1. Supertree Construction

The quickest method of constructing a supertree in clann is to use the command nj, which results in a NJ tree calculated from an average consensus distance matrix to be saved to the file "NJ-tree.ph" and to be displayed on screen (**Fig. 7.11**).

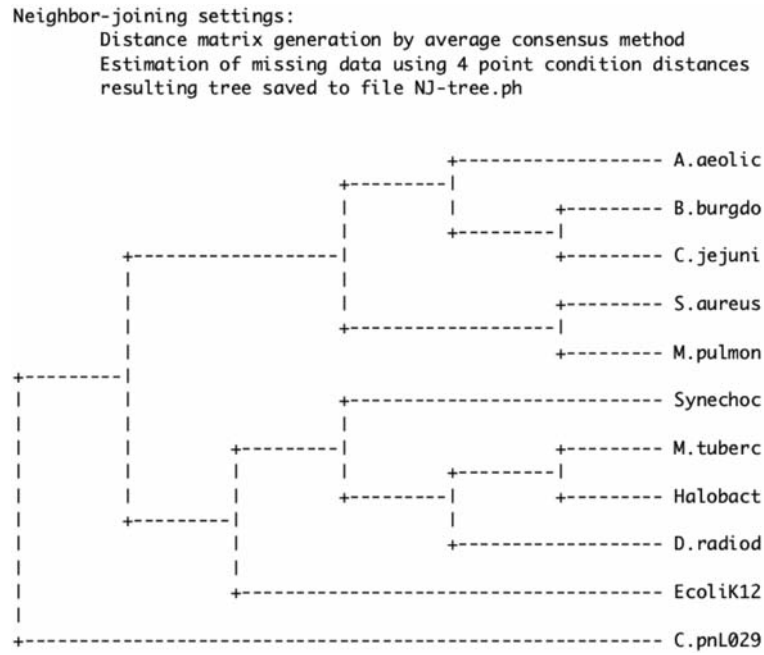


Fig. 7.11. Output generated by the neighbour-joining (nj) command.

It is important to note that the trees displayed by clann are all unrooted. In this dataset *Halobacterium* (Halobact) is the obvious choice as an outgroup when displaying the best supertree in external phylogeny viewers, as it is the only Archaea in the dataset.

By default clann uses the MSSA (DFIT) algorithm when searching for the best supertree. To carry out a simple heuristic search of tree space for the best supertree, type: hs The resulting tree will be saved to a file called "Heuristic_result.txt" and also displayed on screen (**Fig. 7.12**).

Comparing the tree from the NJ algorithm to this tree reveals that they are not the same: for instance in the NJ tree *Mycobacterium tuberculosis* (M.tuberc) is most closely related to *Halobacterium* (Halobact), however, in this tree *Deinococcus radiodurans* (D.radiod) is its closest relative.

To try to resolve to differences between these two trees we can carry out a bootstrapped search of supertree space. In this case we will use the command:

```
boot hsreps=1
```

This tells clann to carry out the bootstrap search (100 times by default) but for each replicate only carry out the heuristic search once (the default is 10). This is to speed up the time taken to do the analysis for the purposes of the example.

```
Heuristic Search settings:
        Criterion = Most Similar Supertree (dfit)
        Heuristic search algorithm = Sub-tree Pruning and Regrafting (SPR)
        Maximum Number of Steps (nsteps) = 3
        Maximum Number of Swaps (maxswaps) = 1000000
        Number of repetitions of Heuristic search = 10
        Weighting Scheme = comparisons
        Starting trees = neighbor-joining tree from Average consensus distances
        Missing data estimated using  4 point condition distances
        Output file = Heuristic_result.txt
==========
Number of topologies tried: 7773

                                          +-------------- B.burgdo
                              +-----------|
                              |           +-------------- A.aeolic
+-----------------------------|
|                             |           +-------------- M.pulmon
|                             +-----------|
|                                         +-------------- C.jejuni
|
|                                         +-------------- EcoliK12
|             +-----------------------------|
|             |                           +-------------- Synechoc
|-------------|
|             |             +------------------------------ Halobact
|             +-------------|
|                           |           +-------------- M.tuberc
|                           +-----------|
|                                       +-------------- D.radiod
|
|                                       +-------------- C.pnL029
+---------------------------------------|
                                        +-------------- S.aureus

Supertree 1 of 1 score = 203.613358

Time taken: 1 Minutes 1 second
```

Fig. 7.12. Output generated by clann using the heuristic search (hs) command under the most similar supertree algorithm (MSSA) criterion.

This analysis returns the results shown in **Fig. 7.13**.

There is obviously very little support for any relationships in this tree, but we can further investigate to see if the phylogenetic signal within the source trees is any better than random noise using the YAPTP test. This test creates randomised versions of the source trees and finds the best supertrees for these randomised datasets. This is carried out 100 times and the scores of the best supertrees for the randomised data are displayed along with the score of the best supertree for the original data. If the phylogenetic signal of the source trees is better than random noise, the score of the best supertree should lie well outside the distribution of scores from the randomised data. The command yaptp returns the results shown in **Fig. 7.14**.

As we know from the hs search the score of the best supertree is 203.6 (using the DFIT criterion), then we can say that this score lies within the distribution of random supertrees and leads us to suspect that the overall phylogenetic information in this dataset is no better than random.

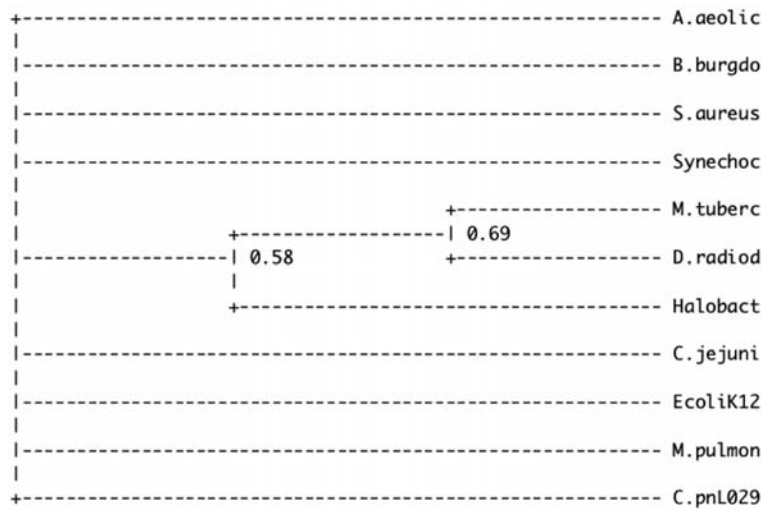


Fig. 7.13. The output generated by clann using the bootstrap (boot) command, detailing the support (or lack thereof) in this dataset.

```
Results as follows:


201.83 - 203.37 |==== (2)
203.38 - 204.91 |==== (2)
204.92 - 206.45 |============ (6)
206.46 - 207.99 |========================== (13)
208.00 - 209.53 |============================ (14)
209.54 - 211.07 |================================ (16)
211.08 - 212.61 |==================================== (18)
212.62 - 214.15 |================================= (17)
214.16 - 215.69 |============ (6)
215.70 - 217.23 |============== (7)


Moments of the Distribution:

        Mean = 208.666380
        Variance = 454.273710
        Standard Deviation = 21.313698
        Skewness = -9.417924
        Standard deviation of skewness = 0.243733


Time taken: 8 Minutes 50 seconds
```

Fig. 7.14. The output generated by clann using the “yet another permutation-tail-probability” (YAPTP) test.

The second dataset from within the Gammaproteobacteria represents a group of organisms that are nearly at the tips of the prokaryotic tree of life. Follow the same procedure with the corresponding file (10taxonfundamentals.ph.txt) to see if the same conclusion holds for this group.

4. Notes

1. Clann can be used to transform nexus formatted tree files into newick-formatted files. This is done by executing the nexus file as normal and then using the command: showtrees savetrees=yes. It is also possible to set the name of the file to which the trees are saved, and to stop clann from displaying a graphical representation of each source tree while this is done.
2. Clann can be told only to read the first few characters of each taxa name when reading the source trees into memory. This is useful when it is necessary to have unique identifiers (for instance gene IDs) on the source trees. The option maxnamelen in the exe command sets this value. If the names are not fixed widths, maxnamelen=delimited tells clann to look for a dot (.) specifying the end of the taxon ID in the trees. For instance using exe maxnamelen=delimited on this tree:(apple.00121, (orange.1435,lemon.3421), pear.1032);- results in clann ignoring the numbers after the dots in the taxa names.
3. The equals sign (=), hyphen (-) and space ( ) are special characters in clann and by default cannot be used in filenames to be read by clann. If a filename contains one of these characters, clann can only read the name of the file properly by putting the name in inverted commas. For example: exe "my -file.txt. "
4. The first command that you should run if you do not know what to do is help. This will display the list of the commands that are available. Calling any of the commands followed by a question mark (for instance, hs ?) will display the options and defaults associated with that command.
5. The command ! runs a shell terminal on Unix and Mac operating systems allowing system commands can be run without having to quit clann.
6. Clann can assess supertrees created using other programs. Using the usertrees command, clann will read in the file specified and assess all the trees it contains. The best supertree found in the file is displayed.

7. All commands in clann should be written completely in lowercase, typing the command boot is not the same as Boot and only the first will be recognised as a valid command.
8. Heuristic and exhaustive searches of supertree space can be interrupted using the key combination "Control-c." This allows the user to specify if they wish to stop the search now and display the best tree found so far. If this is done during the random sampling phase of a heuristic search, it will allow the user to move straight to the heuristic search without completing the random sampling.
9. Users can assess different configurations of their data by excluding (or including) certain source trees from subsequent commands using the excludetrees and includetrees commands. Source trees can be selected based on their name, the taxa they contain, their size (number of taxa they contain) or their score when compared to a supertree.
10. Individual (or multiple) taxa can be pruned from the source trees using the command deletetaxa. Branch lengths are adjusted to take the deletion of the taxa into account. If the deletion of taxa from a source tree means that there are less than four taxa remaining, that source tree is removed from the analysis. Clann will display the names of the source trees removed if this occurs.

References

1. Aho, A. V., Sagiv, Y., Szymanski, T. G., and Ullman, J. D. (1981) Inferring a tree from lowest common ancestors with an application to the optimization of relational expressions. *SIAM J Comput* **10**, 405–21.
2. Gordon, A. D. (1986) Consensus supertrees: the synthesis of rooted trees containing overlapping sets of laballed leaves. *J Classification* **3**, 335–48.
3. Wilkinson, M., Cotton, J. A., Creevey, C., Eulenstein, O., Harris, S. R., Lapointe, F. J., Levasseur, C., McInerney, J. O., Pisani, D., and Thorley, J. L. (2005) The shape of supertrees to come: tree shape related properties of fourteen supertree methods. *Syst Biol* **54**, 419–31.
4. Liu, F. G., Miyamoto, M. M., Freire, N. P., Ong, P. Q., Tennant, M. R., Young, T. S., and Gugel, K. F. (2001) Molecular and morphological supertrees for eutherian (placental) mammals. *Science* **291**, 1786–9.
5. Beck, R. M., Bininda-Emonds, O. R., Cardillo, M., Liu, F. G., and Purvis, A. (2006) A higher-level MRP supertree of placental mammals. *BMC Evol Biol* **6**, 93.
6. Dagan, T., and Martin, W. (2006) The tree of one percent. *Genome Biol* **7**, 118.
7. Creevey, C. J., Fitzpatrick, D. A., Philip, G. K., Kinsella, R. J., O'Connell, M. J., Pentony, M. M., Travers, S. A., Wilkinson, M., and McInerney, J. O. (2004) Does a tree-like phylogeny only exist at the tips in the prokaryotes? *Proc R Soc Lond B Biol Sci* **271**, 2551–8.
8. McInerney, J. O., and Wilkinson, M. (2005) New methods ring changes for the tree of life. *Trends Ecol Evol* **20**, 105–7.
9. Pollard, D. A., Iyer, V. N., Moses, A. M., and Eisen, M. B. (2006) Widespread discordance of gene trees with species tree in Drosophila: evidence for incomplete lineage sorting. *PLoS Genet* **2**, e173.
10. Doolittle, W. F. (1999) Lateral genomics. *Trends Cell Biol* **9**, M5–8.
11. Jain, R., Rivera, M. C., and Lake, J. A. (1999) Horizontal gene transfer among

genomes: the complexity hypothesis. *Proc Natl Acad Sci USA* **96**, 3801–6.

12. Garcia-Vallve, S., Romeu, A., and Palau, J. (2000) Horizontal gene transfer in bacterial and archaeal complete genomes. *Genome Res* **10**, 1719–25.
13. Brown, J. R., Douady, C. J., Italia, M. J., Marshall, W. E., and Stanhope, M. J. (2001) Universal trees based on large combined protein sequence data sets. *Nat Genet* **28**, 281–5.
14. Kim, J., and Salisbury, B. A. (2001) A tree obscured by vines: horizontal gene transfer and the median tree method of estimating species phylogeny. *Pac Symp Biocomput*, 571–82.
15. Dutta, C., and Pan, A. (2002) Horizontal gene transfer and bacterial diversity. *J Biosci* **27**, 27–33.
16. Dagan, T., and Martin, W. (2007) Ancestral genome sizes specify the minimum rate of lateral gene transfer during prokaryote evolution. *Proc Natl Acad Sci USA* **104**, 870–5.
17. Woese, C. R. (2002) On the evolution of cells. *Proc Natl Acad Sci USA* **99**, 8742–7.
18. Doolittle, W. F. (1998) A paradigm gets shifty. *Nature* **392**, 15–6.
19. Pisani, D., Cotton, J. A., and McInerney, J. O. (2007) Supertrees disentangle the chimerical origin of eukaryotic genomes. *Mol Biol Evol* **24**, 1752–60.
20. Hendy, M. D., and Penny, D. (1989) A framework for the quantitative study of evolutionary trees. *Syst Zool* **38**, 297–309.
21. Foster, P. G., and Hickey, D. A. (1999) Compositional bias may affect both DNA-based and protein-based phylogenetic reconstructions. *J Mol Evol* **48**, 284–90.
22. Keane, T. M., Creevey, C. J., Pentony, M. M., Naughton, T. J., and McLnerney, J. O. (2006) Assessment of methods for amino acid matrix selection and their use on empirical data shows that ad hoc assumptions for choice of matrix are not justified. *BMC Evol Biol* **6**, 29.
23. Rokas, A., Williams, B. L., King, N., and Carroll, S. B. (2003) Genome-scale approaches to resolving incongruence in molecular phylogenies. *Nature* **425**, 798–804.
24. Ciccarelli, F. D., Doerks, T., von Mering, C., Creevey, C. J., Snel, B., and Bork, P. (2006) Toward automatic reconstruction of a highly resolved tree of life. *Science* **311**, 1283–7.
25. Philippe, H., Snell, E. A., Bapteste, E., Lopez, P., Holland, P. W., and Casane, D. (2004) Phylogenomics of eukaryotes: impact of missing data on large alignments. *Mol Biol Evol* **21**, 1740–52.
26. Tekaia, F., Lazcano, A., and Dujon, B. (1999) The genomic tree as revealed from whole proteome comparisons. *Genome Res* **9**, 550–7.
27. Snel, B., Huynen, M. A., and Dutilh, B. E. (2005) Genome trees and the nature of genome evolution. *Annu Rev Microbiol* **59,** 191–209.
28. Huson, D. H., and Steel, M. (2004) Phylogenetic trees based on gene content. *Bioinformatics* **20**, 2044–9.
29. Hahn, M. W., De Bie, T., Stajich, J. E., Nguyen, C., and Cristianini, N. (2005) Estimating the tempo and mode of gene family evolution from comparative genomic data. *Genome Res* **15**, 1153–60.
30. Novozhilov, A. S., Karev, G. P., and Koonin, E. V. (2005) Mathematical modeling of evolution of horizontally transferred genes. *Mol Biol Evol* **22**, 1721–32.
31. Lake, J. A., and Rivera, M. C. (2004) Deriving the genomic tree of life in the presence of horizontal gene transfer: conditioned reconstruction. *Mol Biol Evol* **21**, 681–90.
32. Baum, B. R. (1992) Combining trees as a way of combining data sets for phylogenetic inference, and the desirability of combining gene trees. *Taxon* **41**, 3–10.
33. Ragan, M. A. (1992) Matrix representation in reconstructing phylogenetic- relationships among the eukaryotes. *Biosystems* **28**, 47–55.
34. Lapointe, F.-J., and Cucumel, G. (1997) The average consensus procedure: combination of weighted trees containing identical or overlapping sets of taxa. *Syst Biol* **46**, 306–12.
35. Lapointe, F. J., and Levasseur, C. (2004) *Phylogenetic Supertrees: Combining Information to Reveal the Tree of Life* (Bininda-Emonds, O. R. P., Ed.), Vol. 4, Kluwer Academic, Dordrecht.
36. Saitou, N., and Nei, M. (1987) The neighbor-joining method: a new method for reconstructing phylogenetic trees. *Mol Biol Evol* **4**, 406–25.
37. Creevey, C. J., and McInerney, J. O. (2005) Clann: investigating phylogenetic information through supertree analyses. *Bioinformatics* **21**, 390–2.

38. Swofford, D. L. (2002) *PAUP*. Phylogenetic Analysis Using Parsimony (*and Other Methods)*. Version 4, Sinauer Associates, Sunderland, Massachusetts.
39. Maddison, D. R., Swofford, D. L., and Maddison, W. P. (1997) Nexus: an extensible file format for systematic information. *Syst Biol* **46**, 590–621.
40. Page, R. D. M. (1996) TREEVIEW: an application to display phylogenetic trees on personal computers. *Comput Appl Biosci* **12**, 357–8.
41. Letunic, I., and Bork, P. (2007) Interactive Tree Of Life (iTOL): an online tool for phylogenetic tree display and annotation. *Bioinformatics* **23**, 127–8.

Chapter 8

Detecting Signatures of Selection from DNA Sequences Using Datamonkey

Art F.Y. Poon, Simon D.W. Frost, and Sergei L. Kosakovsky Pond

Abstract

Natural selection is a fundamental process affecting all evolving populations. In the simplest case, positive selection increases the frequency of alleles that confer a fitness advantage relative to the rest of the population, or increases its genetic diversity, and negative selection removes those alleles that are deleterious. Codon-based models of molecular evolution are able to infer signatures of selection from alignments of homologous sequences by estimating the relative rates of synonymous (dS) and non-synonymous substitutions (dN). Datamonkey (http://www.datamonkey.org) provides a user-friendly web interface to a wide collection of state-of-the-art statistical techniques for estimating dS and dN and identifying codons and lineages under selection, even in the presence of recombinant sequences.

Key words: Positive selection, adaptive evolution, dN and dS estimation, HyPhy, phylogenetic analysis, maximum likelihood inference, parallel algorithms, web service.

1. Introduction

Natural selection plays a pivotal role in shaping the genetic variation of populations and driving the differentiation of biological taxa. The study of causes and mechanisms of molecular adaptation is one of the fundamental goals of evolutionary biology, with numerous important applications.

Most current techniques for the inference of selection from protein-coding sequences are based on the following observation: a *non-synonymous* (or replacement) substitution in a protein-coding sequence changes the primary sequence of the encoded protein and is more likely to influence the fitness of an organism than a random *synonymous* substitution that leaves the amino acid

David Posada (ed.), *Bioinformatics for DNA Sequence Analysis,* Methods in Molecular Biology 537

DOI 10.1007/978-1-59745-251-9_8

sequence unchanged. If non-synonymous mutations at a particular codon site in the sequence have a negligible effect on the function or expression of the protein (and hence on its fitness), then the rate of non-synonymous substitutions (dN) should be comparable to the rate of synonymous substitutions (dS), and the site evolves *neutrally*. An excess of non-synonymous substitutions ($dN>dS$) can be interpreted as *positive selection* – suggestive that replacement substitutions increase fitness. A paucity of replacement changes ($dN<dS$) indicates that negative selection is working to remove such substitutions from the gene pool.

The ratio $\omega = dN/dS$ (sometimes also denoted K_A/K_S) has seen wide adoption as a measure of selective pressure *(1, 2)*. Datamonkey (http://www.datamonkey.org) *(3)* is one of the many available software tools for estimating ω (Datamonkey actually reports dN– dS, *see* **Note 1**) using a variety of evolutionary models, with several unique advantages. Datamonkey has an intuitive and streamlined interface that provides easy access to complex, state-of-the-art evolutionary models. Complex models are quickly fitted using a remote computer cluster; an analysis that would otherwise take hours to run on a conventional desktop computer will finish in minutes on a cluster. The collection of available methods is constantly updated as novel techniques are published, obviating the need for a practicing scientist to keep pace with new methodological advances and to install and learn how to use a plethora of software packages. Finally, Datamonkey can analyze selection in the presence of recombination – something few other publicly accessible programs can currently do.

Datamonkey uses the HyPhy package *(4)* as its computational engine. All of the selection analyses implemented in Datamonkey, as well as a number of other analyses, can also be carried out directly in HyPhy. For an in-depth discussion of the methods and a tutorial on how sequences can be analyzed for selection in HyPhy, we direct the interested reader to ref. *(5)*.

Currently, Datamonkey may be used to address the following questions:

- Which codon sites in the alignment are subject to positive or negative selection? SLAC/FEL/REL methods *(6)* can estimate dN and dS at each codon site. More specialized hypotheses can also be tested. For instance, if the alignment contains sequences from multiple individuals (e.g., viruses) then sites subject to positive selection at the level of a population can be identified? (the IFEL method *(7)*).
- At what point in the evolutionary history of sequences did selection occur? The GA Branch method *(8)* can be used to assign values of dN/dS to every branch (lineage) in the phylogenetic tree.

- Does a sequence alignment contain recombinant sequences (GARD *(9)*)? Many traditional selection techniques can be misled by recombination *(10)*, but recombination can be corrected for by identifying non-recombinant fragments in the alignment and reconstructing a phylogenetic tree for each fragment *(9)*.
- Is there evidence of positive selection operating within recombining fragments of the alignment? The PARRIS test *(11)* is used to determine whether a proportion of sites have $dN>dS$ in the context of recombination.

2. Program Usage

To perform a selection analysis, Datamonkey requires an uploaded alignment of at least three homologous coding nucleotide sequences (*see* **Note 2** for browser compatibility). Codon-based methods for estimating dN and dS can be applied to any sequence alignment, but there are several considerations to keep in mind. Ideally, the alignment should represent a single gene or a part thereof (e.g., a subunit), sampled over multiple taxa (e.g., mammalian interferon genes) or a diverse population sample (e.g., Influenza A viruses infecting different individuals; *see* **Note 3**). The number of sequences in the alignment is important: too few sequences will contain too little information for meaningful inference, while too many may take too long to run. At the time of writing this chapter, Datamonkey permits up to 150 sequences for SLAC analyses, 100 for FEL/IFEL analyses, 40 for REL and PARRIS, and 25 for GA-Branch. These numbers are determined by current hardware availability and will be increased in the future (*see* **Section 2.1**). As a rule of thumb, at least ten sequences are needed to detect selection at a single site (SLAC/FEL/IFEL/REL) with any degree of reliability, while as few as four may be sufficient for alignment-wide inference (PARRIS/GA-Branch). The median number of sequences in an alignment submitted to Datamonkey is 19. In addition, comparative methods may be ill suited to study certain kinds of selection (*see* **Note 4**).

It is a good practice to visually inspect your data to make sure that the sequences are aligned correctly. Of course, one can never be sure that an alignment is objectively "correct," but gross misalignments (e.g., sequences that are out of frame) are easy to spot with software that provides a graphical visualization of the alignment, such as HyPhy *(4)*, Se-Al (http://tree.bio.ed.ac.uk/software/seal/), or BioEdit (http://www.mbio.ncsu.edu/BioEdit/bioedit.html). You should verify that the alignment is in frame, i.e., that it does not contain stop codons, including premature stop codons, indicative of a frame shift, e.g., due to misalignment, or a

non-functional coding sequence, and the terminal stop codon. Your alignment should exclude any non-coding region of the nucleotide sequence, such as introns or promoter regions, for which existing models of codon substitution would not apply. When coding nucleotide sequences are aligned directly, frame-shifting (i.e., not in multiples of 3) gaps may be inserted, since the alignment program often does not take the coding nature of the sequence into account. Therefore it is generally a good idea to align translated protein sequences and then map them back onto constituent nucleotides. Datamonkey will perform a number of checks when it receives coding sequences and report all problems it encounters (*see* **Section 2.1**).

Since Datamonkey uses the HyPhy package as its processing engine, it will accept files containing alignments in FASTA, PHYLIP, MEGA, and NEXUS formats.

If the alignment contains identical sequences, Datamonkey will discard all but one of the duplicate sequences before proceeding. This is done to speed up the analyses because identical sequences do not contribute any information to the likelihood inference procedure (except via base frequencies), but the computational complexity of phylogenetic analyses grows with the number of sequences.

Finally, Datamonkey may rename some of the sequences to conform to HyPhy naming conventions for technical reasons (all sequence names must be valid identifiers, e.g. they cannot contain spaces). This is done automatically and has no effect on the subsequent analyses.

2.1. Common Issues When Preparing the Data for Datamonkey

2.1.1. Non-text Files

Datamonkey expects sequence alignments to be uploaded as text files. Any other format (Word, RTF, PDF) will not be recognized and must be converted into plain text prior to submission.

2.1.2. Non-standard Characters in the Alignment

BioEdit may use the tilde (~) character to denote a gap. The dot (.) character is sometimes used as "match the first sequence" character and sometimes as the gap character. Datamonkey will accept IUPAC nucleotide characters (ACGT/U and ambiguity characters) and "?," "X," "N," or "—" for gap or missing data (Datamonkey is not case sensitive). All other characters in sequence data will be skipped and could result in frame shifts, which will be reported upon upload.

2.1.3. Uploading an Amino-Acid Alignment

Datamonkey employs codon models, which require the knowledge of silent substitutions, lost upon translation to amino-acids.

2.1.4. Termination Codons

Datamonkey will reject any alignments that contains stop codons, even if the stop codon is at the end of the sequence (i.e. if it is a proper termination codon). Please strip all stop codons out of the alignment prior to uploading it (the HyPhy standard analysis Data File Tools:CleanStopCodons.bf can do this by replacing all stop codons with indels).

2.1.5. Alignments That Are Too Gappy

If an alignment contains more than 50% of indels, it may not be properly processed (e.g., it could be read as a protein alignment, depending on the alignment format).

2.1.6. Alignments That Are Too Large

If your alignment exceeds the size currently allowed by Datamonkey, consider running your analysis locally in HyPhy. A detailed discussion of how HyPhy can be used for that purpose can be found in ref. *(5)*

2.1.7. Incorrect Genetic Code

If the genetic code is mis-specified (e.g., the mitochondrial code is applied to nuclear sequences), valid alignments may fail to upload and if they do, then the results may be compromised (because codons are mistranslated). Make sure the correct genetic code is selected on the data upload page.

3. Examples

To demonstrate a typical workflow with Datamonkey (http://www.datamonkey.org), we begin by analyzing 21 sequences of the H5N1 Influenza A virus hemagglutinin gene, available for download in FASTA format from http://www.datamonkey.org/data/Flu.fasta (download this alignment to a text file on your computer). Hemagglutinin is a viral protein expressed on the surface of influenza virions and responsible for binding to the sialic acid receptors of host cells. This protein is heavily targeted by the immune response of the host. Within-gene recombination in influenza is thought to be rare, hence we will proceed with the assumption that a single phylogeny is adequate.

3.1. Upload the Alignment

The front page of http://www.datamonkey.org includes a link to the data upload page (**Fig. 8.1**), either in the tool bar at the top of the page or via a large graphical button at the bottom of the page.

3.1.1. Select the Alignment File

Click on the "Browse" button, located next to the field labeled "Choose a sequence file:" to use your browser's interactive window to locate the file on your computer.

3.1.2. Choose the Genetic Code

Datamonkey can interpret codons using one of the 12 standard genetic codes, but it will employ the "Universal" genetic code by default, which is appropriate for Influenza A.

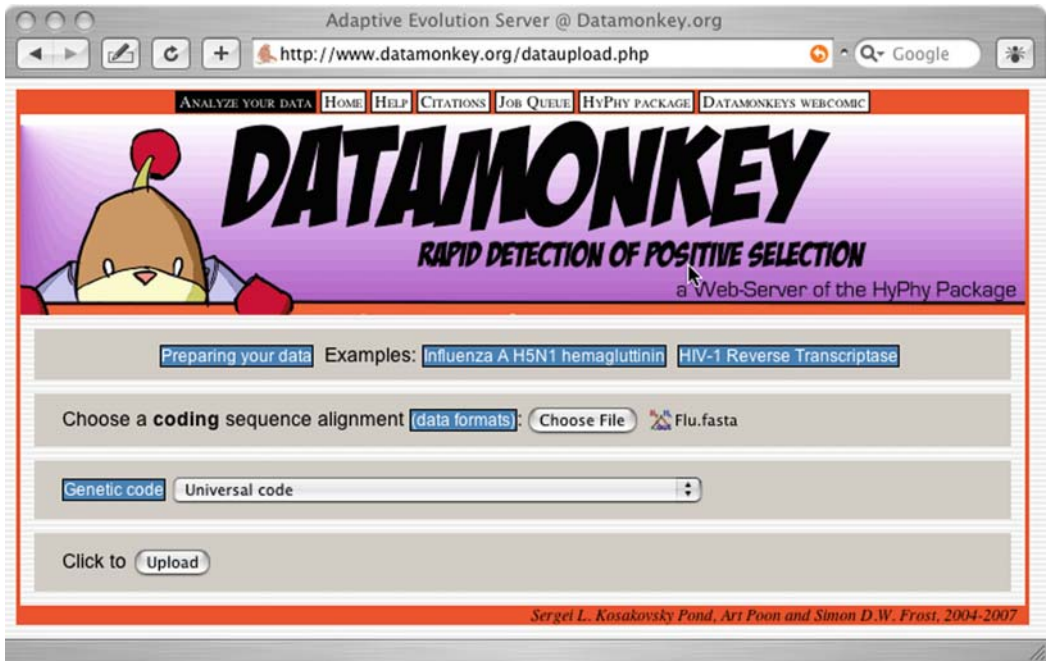


Fig. 8.1. Datamonkey data upload page.

3.1.3. Examine the Uploaded File

Upon a successful upload, Datamonkey reports some basic statistics on the alignment, including the number of sequences, columns (codon sites) and partitions (for alignments with recombinant sequences, there will be multiple partitions – one for each non-recombinant fragment), base frequencies, and an amino-acid translation of the alignment – the PDF version is handy for an at-a-glance sequence consensus and minor variants report (**Fig. 8.2**). You can also verify that the alignment was uploaded correctly by selecting a sequence (using the drop-down menu labeled "BLAST your sequences?") to BLAST against the NCBI non-redundant nucleotide sequence database.

3.1.4. Phylogeny

Datamonkey will automatically detect whether a tree topology (or multiple topologies for recombinant data) is present in the file, e.g., the TREES block in a NEXUS-formatted file. If each tree tip can be matched up with a sequence in the alignment, Datamonkey will make this tree available for analysis. If no tree(s) were found in the file (as is the case for Flu.fasta), then Datamonkey can estimate a neighbor joining (NJ, *(12)*) phylogeny from the alignment, using Tamura-Nei *(13)* nucleotide distances. There is empirical evidence that selection analyses are robust to some error in the phylogeny, hence a "quick and dirty" method like NJ should be sufficient in most cases. Click on the "Proceed" button to move on to the analysis setup page.

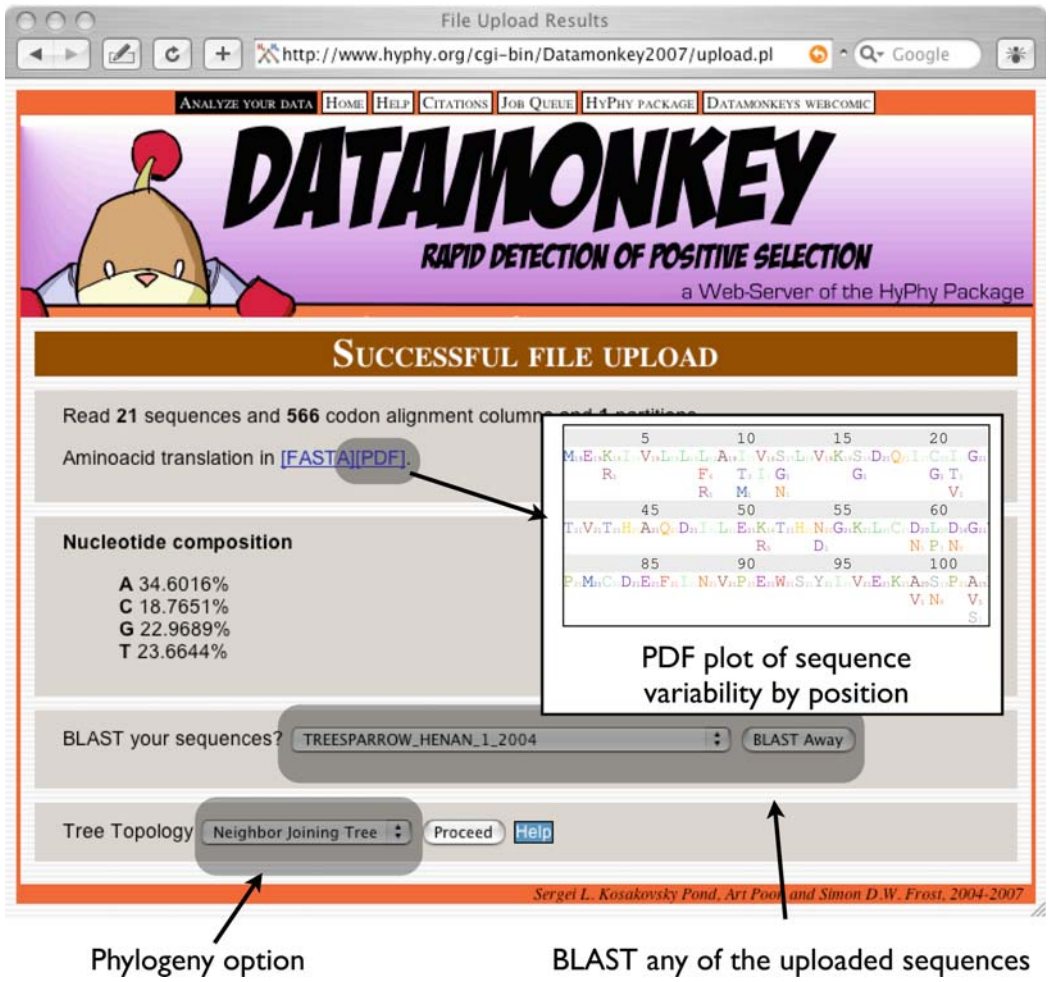


Fig. 8.2. Data upload summary page.

3.2. Analysis Setup Page

The analysis setup page (**Fig. 8.3**) is used to configure all selection analyses available via Datamonkey.

3.2.1. Job Status

The first thing to note is the **Job ID** bar. Each successfully uploaded file will be assigned a random identifier, which can be used to track all the analyses performed on the alignment. Clicking on the [get info] link brings up the status page for this alignment. Nearly every Datamonkey page will have a link to the status page, and we will later discuss how the status page can be used in more detail.

3.2.2. Method

This drop-down menu lists all analyses that can be run on the uploaded alignment. Some of the options may not be present for large and/or recombinant alignments. We will perform SLAC, FEL, and REL analyses (for detecting sites under selection) on Flu.fasta.

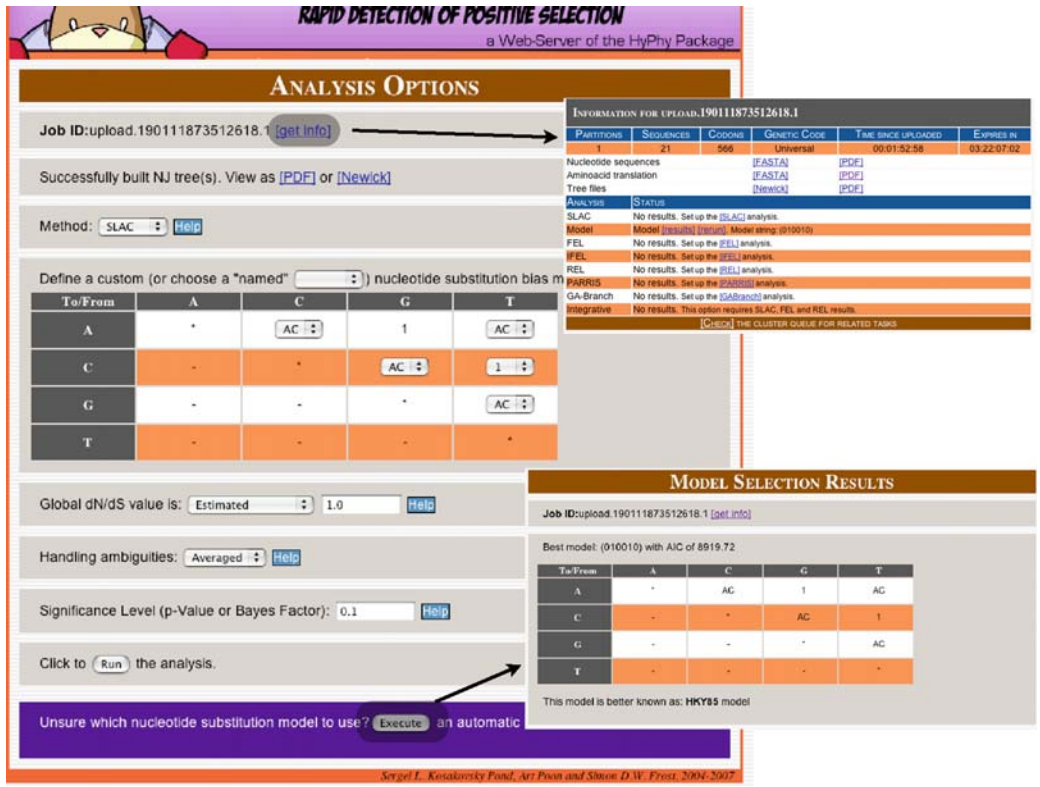


Fig. 8.3. Analysis setup page.

3.2.3. Nucleotide Substitution Bias Model

Each of the methods implemented by Datamonkey makes use of a nucleotide substitution model to estimate the branch lengths and nucleotide substitution biases (such as transition/transversion biases) of the tree from your alignment. Datamonkey can make use of one of the 203 time-reversible nucleotide substitution models. The most general supported time-reversible model (denoted as REV) is comprised of eight free parameters (three nucleotide frequencies + five substitution rates). Four of the most frequently used models (F81, HKY85, TrN93, and REV) are predefined as "named" options.

A parameter-rich model could conceivably overfit a small alignment, while a model that is too simple may lead to biased inference; for this reason, Datamonkey provides an automated tool (link at the bottom of the analysis setup page, *see* **Fig. 8.3**) that will select the best-fitting nucleotide model (*see* **Note 5**) from all 203 reversible models *(14)*. Run the model selection procedure on Flu.fasta and verify that the HKY85 *(15)* model is the best fitting model for influenza hemagglutinin. After the model selection analysis is finished, you can return to the analysis setup page from the model results page by clicking on the [get info] link in the Job ID bar, and then on the link offering to set-up the SLAC analysis.

3.2.4. Analysis Options

There are up to three analysis options (the first two only apply to SLAC, *see* **Note 6**). Each option has a link to the relevant help page, explaining what each settings means. Significance level determines how conservative each method should be, but this option only affects how the results are presented and can be adjusted after the analysis has finished. We begin by submitting a SLAC analysis with the HKY85 model and default analysis options.

3.3. The Job Queue Page

After you submit the analysis by clicking on "Run" button, Datamonkey will display a page with all the jobs currently queued for execution. The newly submitted page analyses are inserted at the end of the queue, and must wait for the jobs in front of it to finish. You may bookmark the queue page in your web browser and return to it later to check on the progress update. Once Datamonkey begins processing your analysis, it will display intermediate progress reports for the task and present a result page when it becomes available.

3.4. SLAC Results Page

All Datamonkey analysis result pages consist of the following three (or four) parts (**Fig. 8.4**).

- *The job ID bar* including the universal [get info] link to bring up the central job summary page.
- *Data and analysis summary.* For SLAC, Datamonkey reports descriptive statistics of the alignment (partitions, codons, and estimated evolutionary tree lengths – note that very long trees can be indicative of misaligned sequences and are flagged as such), the inferred nucleotide substitution biases, log likelihood scores for the fitted models, and the estimate of the alignment-wide dN/dS.
- *Links to more detailed results.* A detailed or graphical output of various analysis specific quantities can be accessed via these links. For SLAC you can, for instance, plot dN-dS across sites, or map the number of inferred synonymous and non-synonymous substitutions to each branch of the tree (**Fig. 8.4**).
- *(SLAC/FEL/IFEL/REL) Summary of selected sites.* Given a specific significance level, all those codons, which the method detected to be under positive or negative selection are reported. This section can be regenerated on the fly for a different significance level (Retabulate).

3.5. Interpreting SLAC Results

The summary output of selected sites lists every site where $dN/dS \neq 1$ with statistical significance (p-value) no greater than the value supplied by the user. The p-value bounds the rate of false positives, e.g., p=0.05 means that up to 5% of neutrally evolving sites may be incorrectly classified as selected (i.e., false positives). The p-value should be taken as a guideline because the statistical properties of

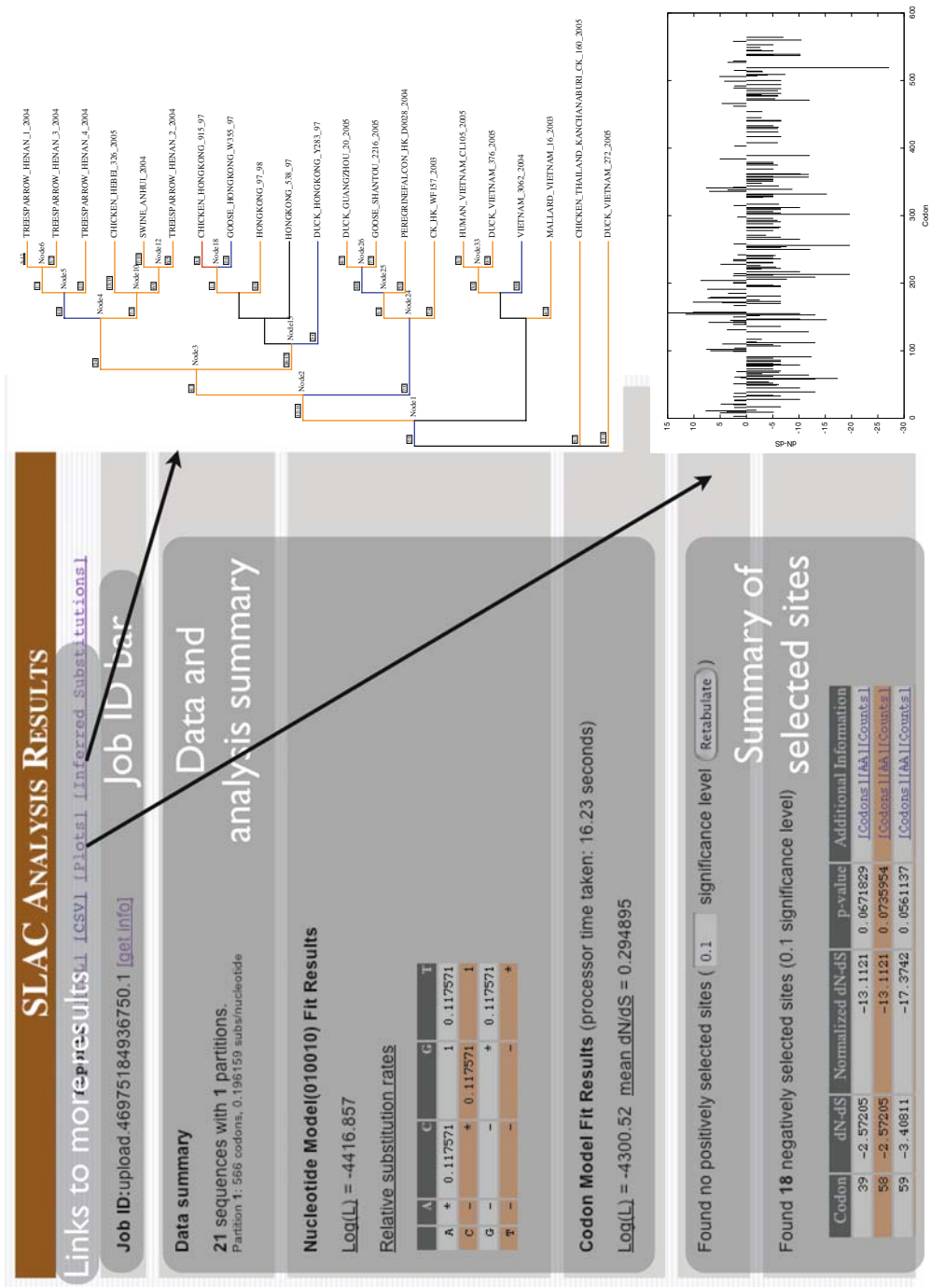


Fig. 8.4. SLAC results page for the influenza analysis.

the test may vary from alignment to alignment. For instance, SLAC tends to be a very conservative test *(6)*, hence the actual rate of false positives can be much lower than the significance level (hence the default of 0.1, *see* **Note 7**). For each codon site, Datamonkey will report the estimated $dN–dS$ (*see* **Note 1**), $dN–dS$ scaled by the total length of the tree (to facilitate direct comparison between different data sets), p-value for the test $dN \neq dS$ at that codon, and links to investigate the inferred mutations at that site.

You may notice that at p=0.1, SLAC reports no positively selected sites. Increase p to 0.2 and retabulate the results to find that four codons (154, 156, 157, 172) have p-values for positive selection in the 0.1–0.2 range (borderline selection). For codon 156, Normalized dN-dS = 14.93. If you now click on the [Counts] link in the Additional Info column, Datamonkey will display the list of inferred substitutions at that codon, showing at least six non-synonymous and zero synonymous substitutions, mapped to five branches of the tree (**Fig. 8.5**). The p-value of 0.12 corresponds to the binomial probability that all six substitutions at that site will be non-synonymous by chance (*see* **Note 8**). SLAC estimates a number of quantities, which enable it to compute this probability (accessible via the detailed HTML report from the SLAC result page). For site 156, its inferred codon composition and relative branch lengths predict that a random substitution is

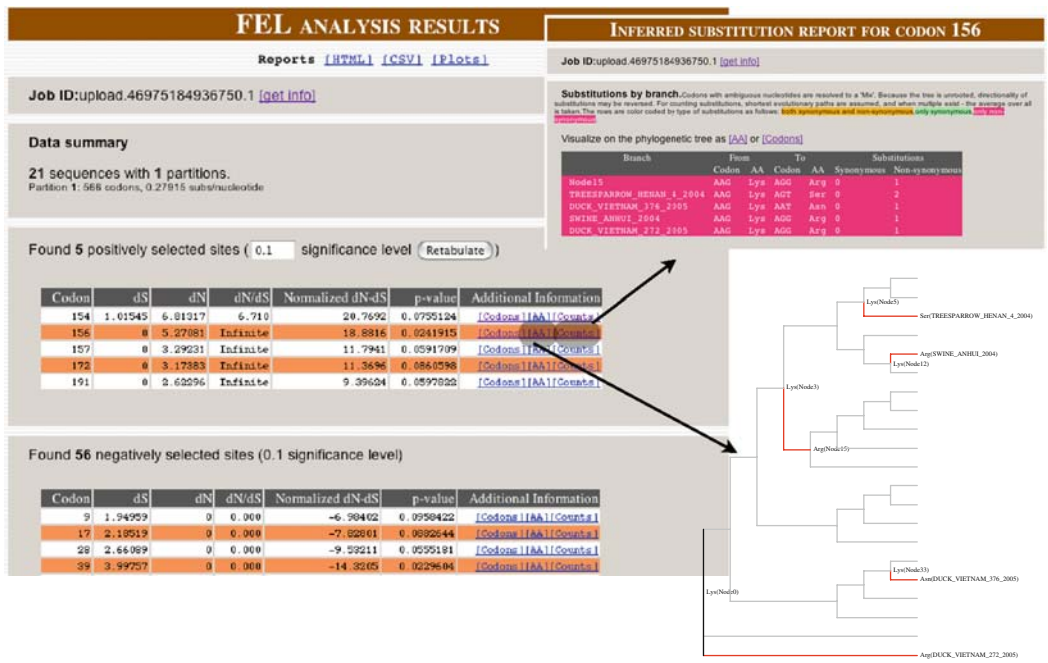


Fig. 8.5. FEL results page for the influenza analysis and substitution maps for codon 156.

synonymous with probability 0.30, assuming neutral evolution, such that the probability of observing 0/6 synonymous substitutions equals $(1–0.3)^6 \approx 0.12$, yielding the *p*-value.

3.6. FEL

Next, return to the analysis setup page (**Fig. 8.2**), via the Job ID bar, and run the FEL analysis with default options (note that if you have run a substitution model selection, Datamonkey will automatically select the appropriate model on the analysis setup page). The FEL result page (**Fig. 8.5**) is similar to the SLAC result page.

Like SLAC, FEL evaluates dS and dN at each site, but instead of basing its inference on the expected and inferred numbers of synonymous and non-synonymous substitutions, FEL directly estimates dN and dS based on a codon-substitution model, and derives the *p*-value for the test $dN \neq dS$ using a likelihood ratio test *(6)*. FEL tends to be quite a bit more powerful than SLAC, e.g., note that all four borderline positively selected SLAC sites, have *p*-value in 0–0.1 for FEL, but it is an order of magnitude more computationally expensive.

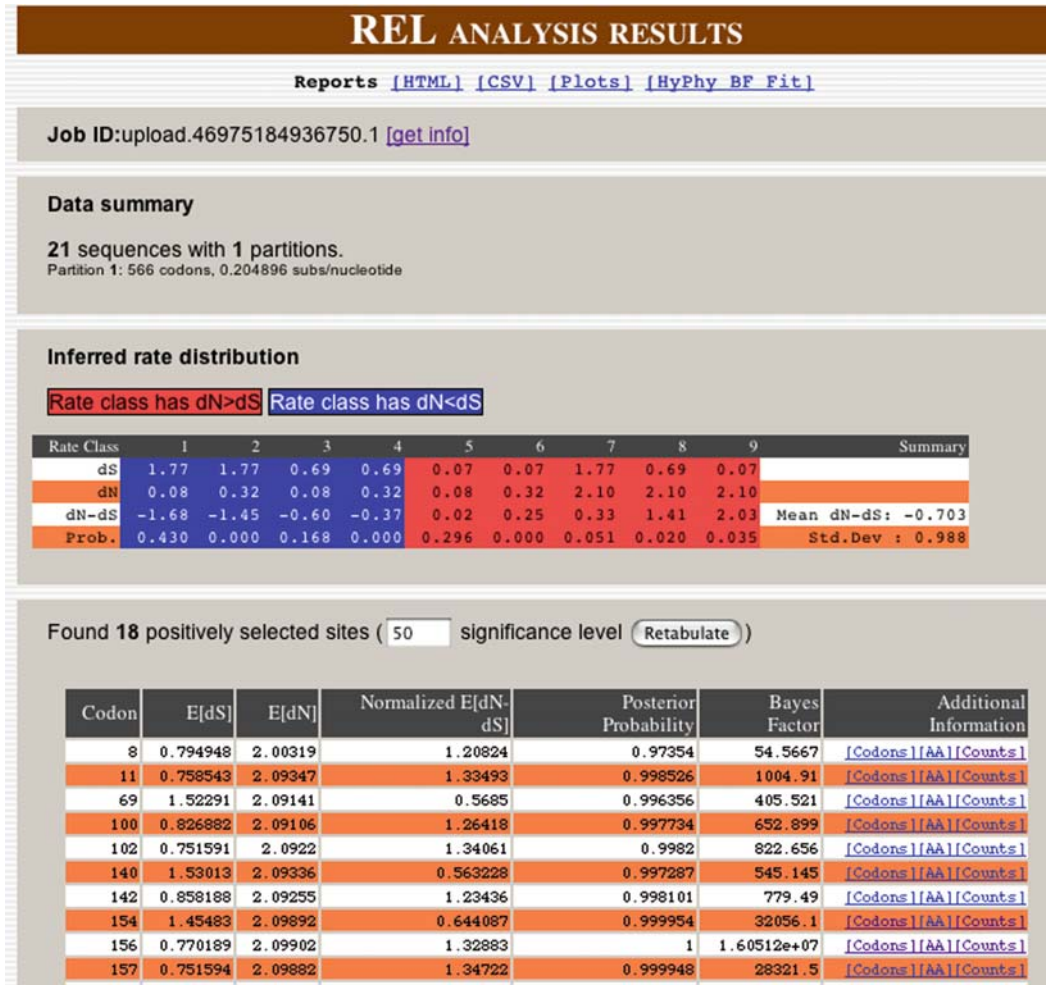
3.7. REL

Run the REL analysis with default options (**Fig. 8.6**). REL is similar to the popular likelihood methods implemented in the PAML package *(16)*, with several important additions (e.g., synonymous rate variation, *see (6)*). Instead of directly estimating dS and dN at each site, REL estimates the parameters for discretized *distributions* of dS and dN (with three rate categories for a total of nine possible rate combinations) from the entire alignment and then infers which of these each site is most likely to have. To decide if a codon has $dN > dS$, the REL method computes a Bayes factor (BF) defined as the ratio of posterior odds of having $dN > dS$ to the prior odds. A large BF suggests that the data lend strong support to the hypothesis that a site is positively selected (as a rule of thumb, 1/BF is similar to the *p*-value). REL tends to be the most powerful of the three tests because it uses the entire alignment to make inferences about rates at each site, but also generally has the highest rate of false positives because the distribution of rates to be fitted must be defined *a priori*, and it may not adequately model the unobserved distribution of rates.

For our example, 18 sites are called positively selected ($dN > dS$). However, REL does not take the magnitude of dN-dS into account, and nearly neutral or invariable sites (such as those from rate classes 5 or 7, **Fig. 8.6**) may count toward the positively selected category.

3.8. Integrative Selection Analysis

Practical statistical inference is never 100% accurate and there will be some false positives and negatives. As our example illustrates, different approaches to inferring the same effect can lead to different inferences. Fortunately, for larger datasets



REL ANALYSIS RESULTS

Reports [HTML] [CSV] [Plots] [HyPhy BF Fit]

Job ID:upload.46975184936750.1 [get info]

Data summary

21 sequences with **1** partitions.
Partition **1**: 566 codons, 0.204896 subs/nucleotide

Inferred rate distribution

Rate class has dN>dS | Rate class has dN<dS

Rate Class	1	2	3	4	5	6	7	8	9	Summary
dS	1.77	1.77	0.69	0.69	0.07	0.07	1.77	0.69	0.07	
dN	0.08	0.32	0.08	0.32	0.08	0.32	2.10	2.10	2.10	
dN-dS	-1.68	-1.45	-0.60	-0.37	0.02	0.25	0.33	1.41	2.03	Mean dN-dS: -0.703
Prob.	0.430	0.000	0.168	0.000	0.296	0.000	0.051	0.020	0.035	Std.Dev : 0.988

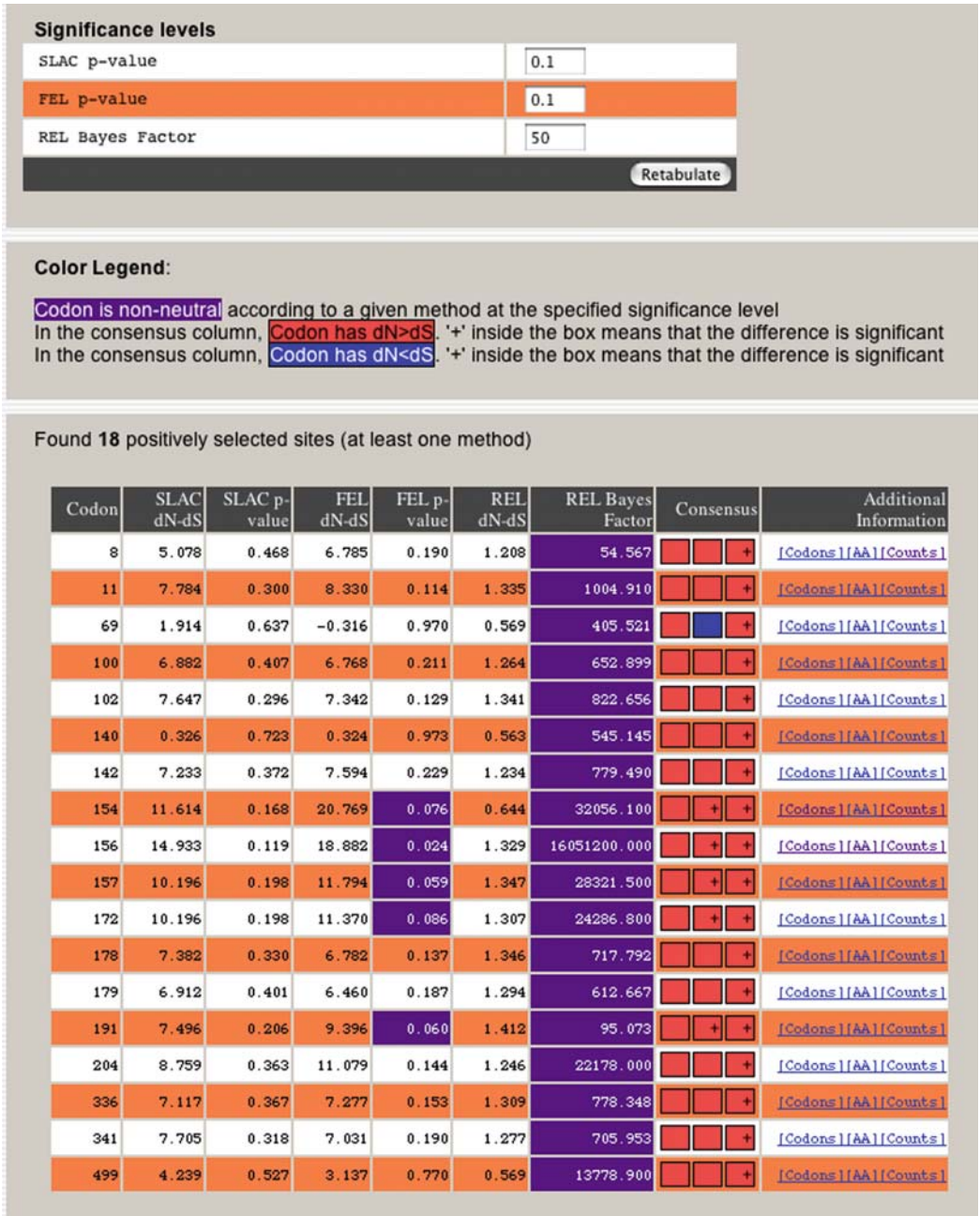
Found **18** positively selected sites (50 significance level Retabulate)

Codon	E[dS]	E[dN]	Normalized E[dN-dS]	Posterior Probability	Bayes Factor	Additional Information
8	0.794948	2.00319	1.20824	0.97354	54.5667	[Codons][AA][Counts]
11	0.758543	2.09347	1.33493	0.998526	1004.91	[Codons][AA][Counts]
69	1.52291	2.09141	0.5685	0.996356	405.521	[Codons][AA][Counts]
100	0.826882	2.09106	1.26418	0.997734	652.899	[Codons][AA][Counts]
102	0.751591	2.0922	1.34061	0.9982	822.656	[Codons][AA][Counts]
140	1.53013	2.09336	0.563228	0.997287	545.145	[Codons][AA][Counts]
142	0.858188	2.09255	1.23436	0.998101	779.49	[Codons][AA][Counts]
154	1.45483	2.09892	0.644087	0.999954	32056.1	[Codons][AA][Counts]
156	0.770189	2.09902	1.32883	1	1.60512e+07	[Codons][AA][Counts]
157	0.751594	2.09882	1.34722	0.999948	28321.5	[Codons][AA][Counts]

Fig. 8.6. REL results page for the influenza analysis.

(e.g., >100 sequences), all three methods tend to agree *(6)*, with the SLAC being the most conservative (i.e., it may miss some selected sites, but will not identify many neutral sites as selected), followed by FEL (more detected sites), and then REL (most detected sites, but highest errors). For smaller datasets, like the example influenza alignment, it is a good idea to run all three methods and compare their results side by side. When the methods corroborate each other's findings, we may be more confident in their inference.

Once SLAC, FEL, and REL have been run, the Integrative Selection Analysis option is enabled on the job status page. The integrative selection page (**Fig. 8.7**) tabulates all codons, which are detected by at least one of the methods, based on the specified significance levels. For the influenza analysis, codons 154, 156, 157, 172, and 191 are supported by FEL and REL, and all have large *dN-dS* values for SLAC with borderline (0.1–0.25) *p*-values.



Significance levels

SLAC p-value	0.1
FEL p-value	0.1
REL Bayes Factor	50

Retabulate

Color Legend:

Codon is non-neutral according to a given method at the specified significance level
In the consensus column, Codon has dN>dS. '+' inside the box means that the difference is significant
In the consensus column, Codon has dN<dS. '+' inside the box means that the difference is significant

Found **18** positively selected sites (at least one method)

Codon	SLAC dN-dS	SLAC p-value	FEL dN-dS	FEL p-value	REL dN-dS	REL Bayes Factor	Consensus	Additional Information
8	5.078	0.468	6.785	0.190	1.208	54.567		[Codons][AA][Counts]
11	7.784	0.300	8.330	0.114	1.335	1004.910		[Codons][AA][Counts]
69	1.914	0.637	-0.316	0.970	0.569	405.521		[Codons][AA][Counts]
100	6.882	0.407	6.768	0.211	1.264	652.899		[Codons][AA][Counts]
102	7.647	0.296	7.342	0.129	1.341	822.656		[Codons][AA][Counts]
140	0.326	0.723	0.324	0.973	0.563	545.145		[Codons][AA][Counts]
142	7.233	0.372	7.594	0.229	1.234	779.490		[Codons][AA][Counts]
154	11.614	0.168	20.769	0.076	0.644	32056.100	+	[Codons][AA][Counts]
156	14.933	0.119	18.882	0.024	1.329	16051200.000	+	[Codons][AA][Counts]
157	10.196	0.198	11.794	0.059	1.347	28321.500	+	[Codons][AA][Counts]
172	10.196	0.198	11.370	0.086	1.307	24286.800	+	[Codons][AA][Counts]
178	7.382	0.330	6.782	0.137	1.346	717.792		[Codons][AA][Counts]
179	6.912	0.401	6.460	0.187	1.294	612.667		[Codons][AA][Counts]
191	7.496	0.206	9.396	0.060	1.412	95.073	+	[Codons][AA][Counts]
204	8.759	0.363	11.079	0.144	1.246	22178.000		[Codons][AA][Counts]
336	7.117	0.367	7.277	0.153	1.309	778.348		[Codons][AA][Counts]
341	7.705	0.318	7.031	0.190	1.277	705.953		[Codons][AA][Counts]
499	4.239	0.527	3.137	0.770	0.569	13778.900		[Codons][AA][Counts]

Fig. 8.7. Integrative selection page for the influenza analysis.

Hence, they are likely to be under diversifying selection. On the other extreme, codon 69 is only identified by REL, while FEL assigned $dN<dS$ (negative selection, but not significant), and SLAC assigns a very large p-value for positive selection; this discordance does not inspire confidence!

It also helps to find confirmatory evidence as to why detected sites may be under positive selection (e.g., based on the structure of the protein or *in vitro* experiments). For example, codons 154, 156, and 157 reside in the previously characterized antigenic domain of hemagluttinin *(17)*, while codon 172 is involved in the formation of a new glycosylation site – an immune evasion mechanism *(18)*.

3.9. Lineage-Specific Selection

If sequences in the alignment come from different selective environments, dN/dS may vary from branch to branch in the phylogenetic tree. Consider 9 HIV-1 envelope sequences isolated from two patients *(19)* where one (the source, five sequences), transmitted the virus to the other (the recipient, four sequences), available for download in NEXUS format from http://www.datamonkey.org/data/env.nex (download this alignment to a text file on your computer). The evolution of HIV envelope protein is strongly influenced by selective pressures exerted by the humoral immune response mounted by the host, and in our example is subject to three potentially different selective environments: the source, the recipient, and the transmission period represented by the branch separating the sequences from each host. We run the GA Branch analysis (*see* **Note 9**) to demonstrate how Datamonkey can be used to segregate all branches into a smaller number of different selective regimes. First, upload the alignment, perform model selection (HKY85 should be selected), and start the GA Branch analysis from the model setup page (**Fig. 8.3**).

The resulting page (**Fig. 8.8**) shows that the data support three rate classes, with a large number of models, which can credibly describe the data (over 2,500 in the 95% confidence set). The summary table for model-averaged dN/dS shows statistical description for estimated dN/dS at each branch. For instance, Node1, which is the internal branch (*see* tree plot in **Fig. 8.8**) representing the transmission event, has $dN>dS$ with high degree (over 0.999 probability support) of confidence, as also evidenced by the histogram of dN/dS tabulated from all credible models (weighted appropriately, *see (8)*). The tree plot is a useful visual representation of the results: each tree branch is labeled with percent support for positive selection along that branch (four branches have over 95% support), while the coloring reflects the selection pattern according to the best fitting model.

As expected, there is evidence of complex and variable selective pressures on the envelope of HIV-1 in this sample, with strong support of positive selection along some (but not all) branches representing the evolution in each patient, as well as along the connecting (transmission) branch (*see* **Note 4**).

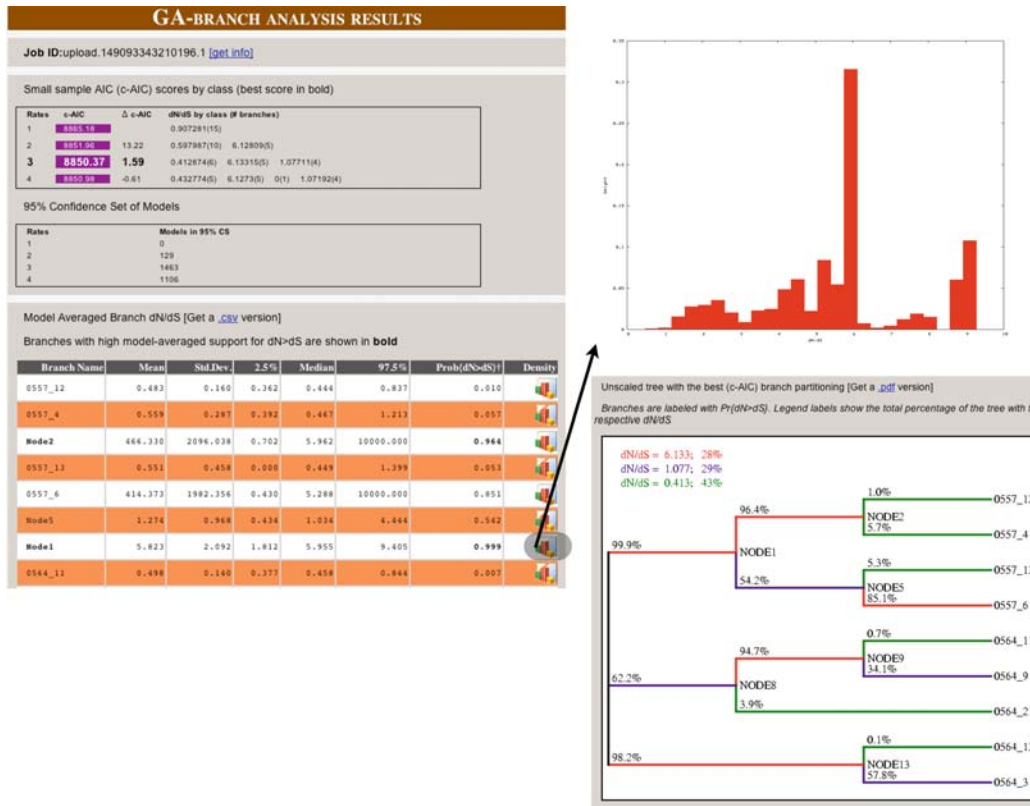


Fig. 8.8. Lineage-specific selection in a two-patient HIV-1 envelope sample.

3.10. Selection in the Presence of Recombination

The presence of recombinants among aligned sequences frequently makes it impossible to represent the evolutionary history of the sample as a single phylogenetic tree. Most traditional selection detection techniques *(20–22)* assume a single phylogeny and can be misled if recombinants are present, e.g., *(9, 10)*. Datamonkey can avoid this shortcoming by first screening the alignment to locate all non-recombinant fragments (partitions) using the GARD method *(9)*, and then allowing each partition to have its own phylogenetic tree. We demonstrate this approach on an alignment of 16 HIV-1 polymerase genes sampled from the Democratic Republic of Congo – an area where multiple divergence variants of HIV are in common circulation *(23)*. When a single host is infected with multiple HIV-1 viruses, this creates the potential for the generation and transmission of recombinant HIV variants because of very high recombination rates in multiply infected cells *(24)*.

Download the file from http://www.datamonkey.org/data/pol.nex. When screened with GARD (http://www.datamonkey.org/GARD), this alignment shows evidence of three

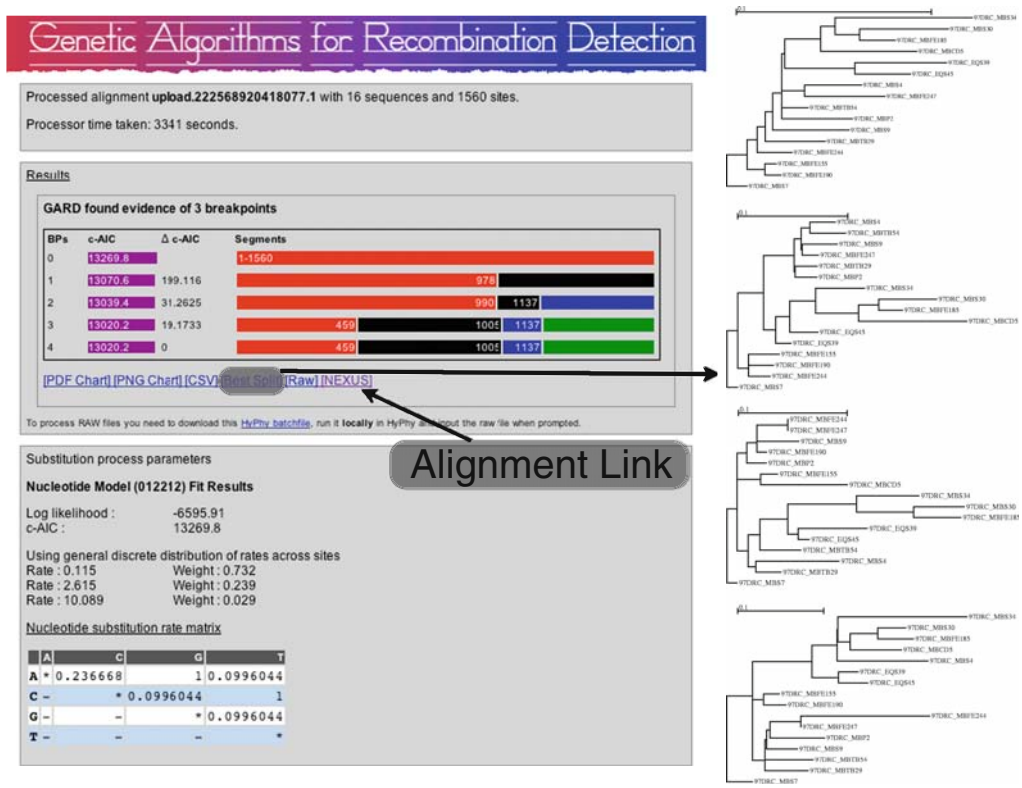


Fig. 8.9. GARD recombination screen on HIV-1 pol alignment.

recombination breakpoints (**Fig. 8.9**). As one of the outputs of a GARD analysis (which we encourage the interested users to run), there is NEXUS format file, which includes the information about all non-recombinant partitions (ASSUMPTIONS block), and the trees inferred for each partition.

This information will be recognized by Datamonkey and incorporated into the analyses (for instance the data upload page will show that four partitions were read and what they were). The rest of the analyses can be carried out exactly as we did previously with a non-recombinant example alignment.

As an exercise in the effect of recombination, compare the multiple-segment analysis with the results on the same alignment assuming a single phylogeny (http://www.datamonkey.org/data/pol-1.nex). In many cases the difference is not dramatic, but noticeable nonetheless (**Fig. 8.10**). For example, both analyses agree of codon 476, the evidence for selection at codon 273 is stronger when multiple trees are allowed, and codon 371 is no longer detected as positively selected when multiple trees are considered.

Single tree

Found **10** positively selected sites (at least one method)

Codon	SLAC dN-dS	SLAC p-value	FEL dN-dS	FEL p-value	REL dN-dS	REL Bayes Factor	Consensus	Additional Information
63	1.882	0.446	0.062	0.977	1.556	159.863		[Codons][AA][Counts]
138	5.106	0.176	1.374	0.582	1.946	527.770		[Codons][AA][Counts]
222	3.093	0.381	1.390	0.490	1.795	5755.920		[Codons][AA][Counts]
272	5.396	0.127	1.798	0.269	1.986	337.832		[Codons][AA][Counts]
273	4.600	0.166	1.832	0.075	1.034	32.112		[Codons][AA][Counts]
344	1.458	0.511	2.855	0.111	1.833	1119.460		[Codons][AA][Counts]
371	4.068	0.131	1.419	0.067	0.377	22.720		[Codons][AA][Counts]
433	5.151	0.203	2.430	0.330	1.987	1106.420		[Codons][AA][Counts]
456	1.649	0.530	3.006	0.106	1.999	2939.530		[Codons][AA][Counts]
476	8.066	0.090	6.985	0.081	2.222	40444.300		[Codons][AA][Counts]

Multiple trees

Found **14** positively selected sites (at least one method)

Codon	SLAC dN-dS	SLAC p-value	FEL dN-dS	FEL p-value	REL dN-dS	REL Bayes Factor	Consensus	Additional Information
35	-0.074	0.685	-0.310	0.928	1.419	380.281		[Codons][AA][Counts]
63	1.226	0.547	0.009	0.998	1.404	1072.030		[Codons][AA][Counts]
138	6.520	0.186	2.022	0.680	1.532	5743.180		[Codons][AA][Counts]
221	-0.654	0.737	-1.176	0.700	1.048	100.643		[Codons][AA][Counts]
222	3.432	0.389	1.955	0.495	1.498	17798.300		[Codons][AA][Counts]
234	1.633	0.597	0.318	0.933	1.496	616.356		[Codons][AA][Counts]
272	5.477	0.178	2.613	0.321	1.604	5779.460		[Codons][AA][Counts]
273	4.845	0.205	2.800	0.083	1.369	143.799		[Codons][AA][Counts]
310	0.005	0.679	-1.219	0.658	0.791	50.488		[Codons][AA][Counts]
344	2.404	0.442	3.082	0.140	1.609	6029.640		[Codons][AA][Counts]
350	3.861	0.259	1.281	0.072	-0.184	2.832		[Codons][AA][Counts]
433	4.006	0.175	1.467	0.322	1.611	4206.950		[Codons][AA][Counts]
456	-0.330	0.688	1.120	0.208	1.650	2519.710		[Codons][AA][Counts]
476	6.274	0.090	4.243	0.093	1.630	32027.300		[Codons][AA][Counts]

Fig. 8.10. Effect of including multiple trees on the detection of sites under selection (HIV-1 pol alignment).

4. Notes

1. The difference is used in place of a more common ratio dN/dS because dS could be 0 for some sites, rendering the ratio infinite.
2. Datamonkey.org has been developed and tested using Mozilla-based browsers (Firefox, Camino) and the Safari browser. While every attempt has been made to write standard compliant HTML and JavaScript code, certain compatibility issues may exist (e.g., when using Internet Explorer). Certain features require that JavaScript be enabled.

3. Because comparative methods estimate relative rates of synonymous and non-synonymous substitution, substantial sequence diversity is needed for reliable inference. For example, when Suzuki and Nei *(25)* applied a REL-type method to a very low divergence (one or two substitutions per sequence along a star phylogeny) sample of the Human T-lymphotropic virus (HTLV), they found that the method performed poorly. Yang and colleagues *(26, 27)* have suggested that the total length of the phylogenetic tree should be at least one expected substitution per codon site, but this is merely a guideline, not a requirement. However, sequences that are too divergent could lead to *saturation*, i.e., the inability to reliably infer branch lengths and substitution parameters.
4. For example, comparative methods should not be applied to the detection of selective sweeps (rapid replacement of one allele with a more fit one, resulting in a homogeneous population), unless sequences sampled prior to and following the selective sweep are included in the sample. We refer an interested reader to *(5)* for further insight.
5. The model test procedure is based on repeated likelihood ratio tests between nested models and AIC comparisons for non-nested models. Akaike's Information Criterion – a goodness-of-fit criterion that rewards the model for higher log-likelihood score (logL), but penalizes it for each additional parameter (p) as follows: AIC = –2logL+2p. The model with the lowest AIC explains the data best. *See (28)* for an excellent treatment on model selection in the phylogenetic context.
6. If JavaScript is enabled in your browser, selecting an analysis will automatically hide all the inapplicable options.
7. Readers interested in technical details should *see (29)* for discussion and a practical example of determining an appropriate significance level in the context of detecting sites under selection.
8. The binomial distribution provides a tractable approximation to the expected proportions of synonymous and non-synonymous substitutions at a site assuming no selection (see http://www.math.cornell.edu/~durrett/sg/sgnote0123.pdf for technical details). Our simulation studies (unpublished) suggest that the binomial approximation is adequate for a wide range of scenarios.
9. The GA branch analysis *(8)* considers a large number of potential models, where each of the branches is allocated to one of several selective regimes, evaluates the fitness of each model by its small-sample AIC score and uses genetic algorithms to evolve the population of potential models toward

the best fitting one. The unique benefit of this approach is that the information from all models can be combined (weighted by the credibility of each model) to avoid incorrect statistical inference due to model mis-specification. The main advantage of the GA approach as opposed to the alternative branch-site family *(22)* of tests is that the GA procedure will automatically mine the data in search of selection patterns, whereas in branch-site tests, one has to select "interesting" branches a priori, potentially oversimplifying or misleading the inference procedure *(5)*.

References

1. Nielsen, R., and Yang, Z. (1998) Likelihood models for detecting positively selected amino acid sites and applications to the HIV-1 envelope gene. *Genetics* **148,** 929–36.
2. Yang, Z. H., Nielsen, R., Goldman, N., and Pedersen, A. M. K. (2000) Codon-substitution models for heterogeneous selection pressure at amino acid sites. *Genetics* **155,** 431–49.
3. Kosakovsky Pond, S. L., and Frost, S. D. W. (2005) Datamonkey: rapid detection of selective pressure on individual sites of codon alignments. *Bioinformatics* **21,** 2531–3.
4. Kosakovsky Pond, S. L., Frost, S. D. W., and Muse, S. V. (2005) HyPhy: hypothesis testing using phylogenies. *Bioinformatics* **21,** 676–9.
5. Kosakovsky Pond, S. L., Poon, A. F. Y., and Frost, S. D. W. (2007) *Phylogenetic Handbook* (Lemey, P., and Pybus, O., Eds.), Estimating selection pressures on alignments of coding sequences (in press; preprint available at http://www.hyphy.org/pubs/hyphybook2007.pdf), Cambridge University Press: Cambridge.
6. Kosakovsky Pond, S. L., and Frost, S. D. W. (2005) Not so different after all: a comparison of methods for detecting amino-acid sites under selection. *Mol Biol Evol* **22,** 1208–22.
7. Kosakovsky Pond, S. L., Frost, S. D. W., Grossman, Z., Gravenor, M. B., Richman, D. D., and Brown, A. J. L. (2006) Adaptation to different human populations by HIV-1 revealed by codon-based analyses. *PLoS Comput Biol* **2,** e62.
8. Kosakovsky Pond, S. L., and Frost, S. D. W. (2005) A genetic algorithm approach to detecting lineage-specific variation in selection pressure. *Mol Biol Evol* **22,** 478–85.
9. Kosakovsky Pond, S. L., Posada, D., Gravenor, M. B., Woelk, C. H., and Frost, S. D. W. (2006) Automated phylogenetic detection of recombination using a genetic algorithm. *Mol Biol Evol* **23,** 1891–901.
10. Shriner, D., Nickle, D. C., Jensen, M. A., and Mullins, J. I. (2003) Potential impact of recombination on sitewise approaches for detecting positive natural selection. *Genet Res* **81,** 115–21.
11. Scheffler, K., Martin, D. P., and Seoighe, C. (2006) Robust inference of positive selection from recombining coding sequences. *Bioinformatics* **22,** 2493–9.
12. Saitou, N., and Nei, M. (1987) The neighbor-joining method: a new method for reconstructing phylogenetic trees. *Mol Biol Evol* **4,** 406–25.
13. Tamura, K., and Nei, M. (1993) Estimation of the number of nucleotide substitutions in the control region of mitochondrial DNA in humans and chimpanzees. *Mol Biol Evol* **10,** 512–26.
14. Kosakovsky Pond, S. L., and Frost, S. D. W. (2005) A simple hierarchical approach to modeling distributions of substitution rates. *Mol Biol Evol* **22,** 223–34.
15. Hasegawa, M., Kishino, H., and Yano, T. A. (1985) Dating of the human ape splitting by a molecular clock of mitochondrial DNA. *J Mol Evol* **22,** 160–74.
16. Yang, Z. H. (1997) PAML: a program package for phylogenetic analysis by maximum likelihood. *Comput Appl Biosci* **13,** 555–6.
17. Caton, A. J., Brownlee, G. G., Yewdell, J. W., and Gerhard, W. (1982) The antigenic structure of the influenza virus A/PR/8/34 hemagglutinin (H1 subtype). *Cell* **31,** 417–27.

18. Perdue, M. L., and Suarez, D. L. (2000) Structural features of the avian influenza virus hemagglutinin that influence virulence. *Vet Microbiol* **74,** 77–86.
19. Frost, S. D. W., Little, S. J., Kosakovsky Pond, S. L., Chappey, C., Liu, Y., Wrin, T., Petropoulos, C. J., and Richman, D. D. (2005) Characterization of HIV-1 envelope variation and neutralizing antibody responses during transmission of HIV-1 subtype B. *J Virol* **79,** 6523–7.
20. Nielsen, R., and Yang, Z. H. (1998) Likelihood models for detecting positively selected amino acid sites and applications to the {HIV-1} envelope gene. *Genetics* **148,** 929–36.
21. Suzuki, Y., and Gojobori, T. (1999) A method for detecting positive selection at single amino acid sites. *Mol Biol Evol* **16,** 1315–28.
22. Yang, Z., and Nielsen, R. (2002) Codon-substitution models for detecting molecular adaptation at individual sites along specific lineages. *Mol Biol Evol* **19,** 908–17.
23. Vergne, L., Peeters, M., Mpoudi-Ngole, E., Bourgeois, A., Liegeois, F., Toure-Kane, C., Mboup, S., Mulanga-Kabeya, C., Saman, E., Jourdan, J., Reynes, J., and Delaporte, E. (2000) Genetic diversity of protease and reverse transcriptase sequences in non-subtype-B human immunodeficiency virus type 1 strains: evidence of many minor drug resistance mutations in treatment-naive patients. *J Clin Microbiol* **38,** 3919–25.
24. Posada, D. (2002) Evaluation of methods for detecting recombination from DNA sequences: empirical data. *Mol Biol Evol* **19,** 708–17.
25. Suzuki, Y., and Nei, M. (2004) False-positive selection identified by ML-based methods: examples from the sig1 gene of the diatom thalassiosira weissflogii and the tax gene of a human T-cell lymphotropic virus. *Mol Biol Evol* **21,** 914–21.
26. Anisimova, M., Bielawski, J. P., and Yang, Z. H. (2002) Accuracy and power of Bayes prediction of amino acid sites under positive selection. *Mol Biol Evol* **19,** 950–8.
27. Anisimova, M., Bielawski, J. P., and Yang, Z. H. (2001) Accuracy and power of the likelihood ratio test in detecting adaptive molecular evolution. *Mol Biol Evol* **18,** 1585–92.
28. Posada, D., and Buckley, T. R. (2004) Model selection and model averaging in phylogenetics: advantages of Akaike information criterion and Bayesian approaches over likelihood ratio tests. *Syst Biol* **53,** 793–808.
29. Sorhannus, U., and Kosakovsky Pond, S. L. (2006) Evidence for positive selection on a sexual reproduction gene in the diatom genus Thalassiosira (Bacillariophyta). *J Mol Evol* **63,** 231–9.

Chapter 9

Recombination Detection and Analysis Using RDP3

Darren P. Martin

Abstract

Recombination between nucleotide sequences is a major process influencing the evolution of most species on Earth. While its evolutionary value is a matter of quite intense debate, so too is the influence of recombination on evolutionary analysis methods that assume nucleotide sequences replicate without recombining. The crux of the problem is that when nucleic acids recombine, the daughter or recombinant molecules no longer have a single evolutionary history. All analysis methods that derive increased power from correctly inferring evolutionary relationships between sequences will therefore be at least mildly sensitive to the effects of recombination. The importance of considering recombination in evolutionary studies is underlined by the bewildering array of currently available methods and software tools for analysing and characterising it in various classes of nucleotide sequence datasets. Here we will examine the use of some of these tools to derive and test recombination hypotheses for datasets containing a moderate degree of nucleotide sequence diversity.

Key words: Recombination, gene conversion, breakpoints, phylogenetic trees, RDP3.

1. Introduction

Many methods have been developed for the detection and analysis of recombination in nucleotide sequence data (for a reasonably comprehensive list see http://www.bioinf.manchester.ac.uk/recombination/programs.shtml). While most provide some statistical indication of how much evidence of recombination is detectable in a group of nucleotide sequences, some methods will also attempt to provide additional information on recombination. They might, for example, indicate likely recombination breakpoint positions, or provide estimates of the recombination rates needed to account for the number and distribution of recombination signals apparent in a population of sequences.

David Posada (ed.), *Bioinformatics for DNA Sequence Analysis,* Methods in Molecular Biology 537

DOI 10.1007/978-1-59745-251-9_9

At its very core the detection of recombination is remarkably simple. If, for example, three nucleotide sequences are considered, two of them will generally be more closely related to one another than either is to the third. In the absence of recombination it would be expected that every part of all three sequences should reflect this relationship. Detection of recombination is simply the identification of situations where this expectation is provably wrong. Between the very simple but extremely fast "heuristic" pattern recognition methods and the hugely sophisticated but more computationally constrained fully "parametric" methods of recombination detection, there is continuum of approaches that might be used to analyse and characterise recombination.

Given a sample of recombining sequences, what one would ideally like is a computer program that will provide the exact recombination history of every nucleotide position relative to all others in every sequence. Unfortunately none of the available recombination analysis programs are capable of doing this. It is also very improbable that any exact recombination history could ever be reliably inferred from any but the simplest datasets. Given a group of recombining nucleotide sequences the best anyone trying to disentangle their recombinant history could reasonably achieve is a set of consistent recombination hypotheses that describe, in the fewest (or close to the fewest) number of steps, the series of recombination events needed to account for all recombination signals detectable in the sequences. Using the program RDP3 (available from http://darwin.uvigo.es/rdp/rdp.html) *(1)*, a procedure is described here that can be applied to map and characterise a reasonably plausible recombination history for a group of moderately diverse recombining sequences.

2. Program Usage

2.1. Data Files

RDP3 will accept nucleotide sequence alignments in many common formats including FASTA, CLUSTAL, MEGA, NEXUS, GDE, PHYLIP and DNAMAN. Alignments should contain between 3 and 1,000 sequences of up to 32 kilobases in length.

A wide variety of datasets can be productively analysed for recombination as long as care is taken during their assembly. The optimal size of a dataset depends on the degree of sequence diversity present in the dataset. As recombination can only be detected if it occurs in sequences that are not identical to one another, it is important that a dataset contain enough diversity. For datasets with very low degrees of diversity, increasing the lengths of sequences being examined can increase the number of usable variable sites for recombination detection.

It is, however, inadvisable to simply choose the largest dataset possible. The reason for this is that exploratory searches for recombination signals require repetitive statistical testing, with the number of tests performed increasing exponentially with the number of sequences examined and linearly with the lengths of sequences examined. A multiple test correction is therefore absolutely required to prevent false inference of recombination. Unfortunately, guarding against false positives also almost invariably means discarding some evidence of *bone fide* recombination. At a certain point that is dependent on the diversity of the sequences being analysed, the extra recombination signals potentially detectable by increasing either the lengths or numbers of sequences in a dataset are counterbalanced by the increasing severity of multiple testing correction needed to guard against false positives.

2.2. Program Settings

Before screening a nucleotide sequence alignment for recombination it is advisable that you adjust various program settings that influence how RDP3 will search for and analyse recombination signals. All of the program's settings can be changed using the "Options" button at the top of the main program window (**Fig. 9.1**).

Under the "General" tab in the "Analyse Sequences Using:" section (**Fig. 9.2**), you can select the methods you wish to use to detect recombination. It is strongly advised that you use the default selections (RDP, GENECONV and MAXCHI) *(2–4)*. If, however, you are analysing small datasets (<50 sequences) you may wish to also select the CHIMAERA *(5)*, BOOTSCAN *(6)*, 3SEQ *(7)* and SISCAN *(8)* methods. **Table 9.1** gives a brief

Fig. 9.1. Important components of the RDP3 interface.

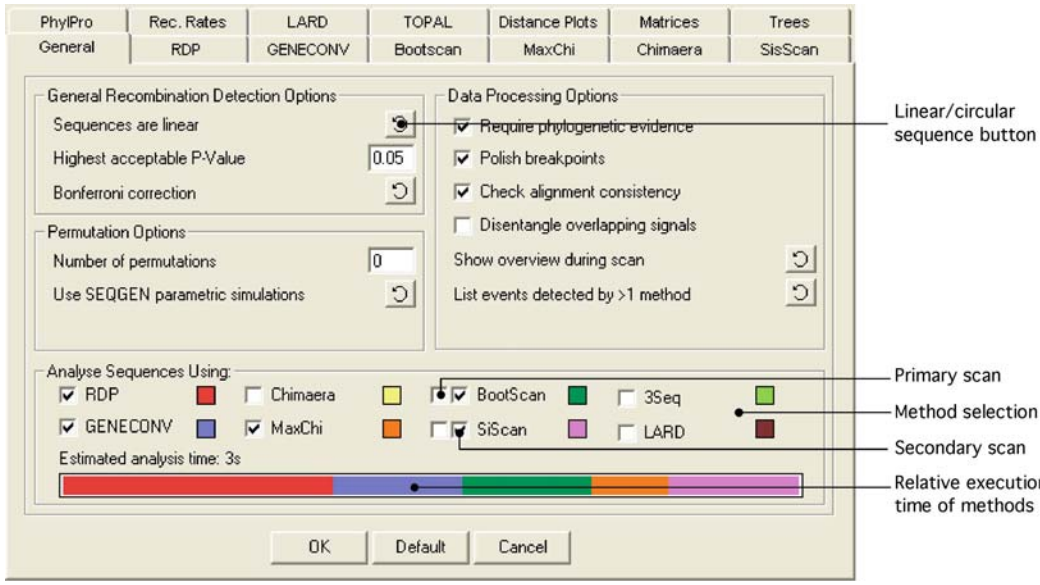


Fig. 9.2. The General settings section of the Options window.

Table 9. 1
Recombination signal detection methods implemented in RDP3

Method	Automated scan [a]	Manual scan[b]	Sliding window[c]	Sites scanned[d]
RDP	+	–	+	V
GENECONV	+	+	–	V
BOOTSCAN	+	+	+	A
MAXCHI	+	+	+	V
CHIMAERA	+	+	+	V
SISCAN	+	+	+	A
3SEQ	+	+	–	V
LARD	–	+	+	A
PHYLPRO	–	–	+	A
TOPAL	–	+	+	A

[a]Can be used to explore for recombination signals without prior knowledge of which sequences are recombinant.
[b]Can be used together with known non-recombinant reference sequences to manually determine whether a "query" sequence is recombinant.
[c]Methods use a sliding window moving along the alignment one or more nucleotides at a time to detect differences between the relatedness of sequences in different parts of the alignment.
[d]V = only variable sites scanned; A = all sites scanned.

description of these methods. Note that there are two possible ways of using the BOOTSCAN and SISCAN methods to detect recombination in an alignment. By default both BOOTSCAN and SISCAN will be used to automatically check recombination signals detected by all other methods but they will not be used to explore for any new signals. You can force their use for exploratory screening by selecting the left box beside the method name, but be warned that the analysis can become very slow if you use these for exploratory screening of large datasets. The RDP, GENECONV, MAXCHI, 3SEQ and CHIMAERA methods will all automatically be used to check recombination signals detected by all other methods regardless of whether they are selected or not. The LARD *(9)* method can only be used to check signals detected by other methods and should only be selected if datasets are very small (<20 sequences). A very rough estimate of the anticipated analysis time is given at the bottom of the options menu so that you can judge whether particular selections are computationally viable.

Under the "Data Processing Options" section of the general tab (**Fig. 9.2**) you can set how RDP3 analyses detected recombination signals during its formulation of a recombination hypothesis. Apart from the "Disentangle overlapping events" option you should use the default settings. If the "Disentangle overlapping events" option is selected the program will attempt to ensure that the recombination hypothesis it derives does not invoke recombination between recombinant sequences with similar mosaics (such as would be identified as relatively unlikely reciprocal recombination events). This setting works well when recombination in the dataset is relatively sparse and there is some evidence of recombination hotspots. However, the algorithm used to disentangle overlapping recombination events can get into a circular loop where it is unable to derive a recombination hypothesis that does not involve, for example, reciprocal recombination. It is therefore recommended that you first try the "disentangle overlapping events" setting and if the analysis appears to get stuck (i.e. takes much longer than the predicted analysis time), you stop it and restart without the setting.

For the method-specific options, the only settings that you should occasionally change from their default values are window and step sizes. The optimal window size will vary slightly from method to method and from dataset to dataset. It is important to note that whereas the RDP, GENECONV, MAXCHI, and CHIMAERA methods only examine variable nucleotide positions in triplets of sequences sampled from the alignment, the BOOTSCAN, SISCAN and LARD methods examine all variable and conserved positions (**Table 9.1**). Also note (a) that the CHIMAERA and MAXCHI windows should be approximately twice as large as the RDP window and (b) that the SISCAN and

BOOTSCAN windows should be approximately the same size. Ideally window sizes should be set small enough to ensure that events involving exchanges of small tracts of sequence (<200 bp) are detectable in the most divergent sequences being examined. The optimal window size to detect a recombination event involving a 200 bp exchange of sequence is 200 for BOOTSCAN and SISCAN and varies for other methods depending on the number of nucleotide differences between the parental viruses in the recombinant region. The MAXCHI and CHIMAERA methods can be set to run with a variable window size that will get bigger and smaller with lower and higher degrees of parental sequence divergence, respectively. Although window size settings can have a substantial impact on the preliminary recombination hypothesis formulated during the automated recombination signal screening stage of analysis, the subsequent (and necessary) phase of manually testing and refining analysis results should largely counteract any "settings biases" that have been introduced.

2.3. Producing a Preliminary Recombination Hypothesis

Once analysis settings have been made and a dataset has been loaded into the program you are ready to begin an automated exploratory search for recombination signals in the dataset. This is accomplished by pressing the "XOver" button (**Fig. 9.1**). Note that there are two main phases in the automated analysis. The first involves the detection of recombination signals in the alignment and the second involves inference of the number and characteristics of unique recombination events that have generated these signals. If the analysis is taking far longer than anticipated you can press the "STOP" button (**Fig. 9.1**). If you prematurely stop the automated analysis, you will be given the analysis results up till the point where the analysis was stopped. From here you can either decide to restart the analysis with different settings or you can move on to looking at the results obtained. If you choose the latter you will be given the opportunity to complete the automated analysis at a later stage.

Once an automated analysis has either been terminated or completed, a set of coloured blocks are presented in a "Schematic sequence display" in the bottom right panel of the program (**Fig. 9.1**). These graphically represent the recombination events that RDP3 has detected. For each sequence in the dataset, the name of the sequence and a coloured strip are displayed. Beneath some of these strips (and corresponding with lightened sections of the coloured strips immediately below the sequence names) are a series of coloured blocks. These blocks each represent a proposed recombination event. If you move the mouse pointer over any of these blocks, information relating to the proposed recombination event will be presented in the "recombination information display" on the top right panel of the screen (**Fig. 9.1**). This information includes possible recombination breakpoints, names of

sequences in the dataset that are most closely related to the parents of the recombinant sequence, the approximated probability values (both corrected and uncorrected for multiple testing) of observing a recombination signal with the same strength without recombination having occurred, the number of sequences in the dataset with similar signals detected by different recombination detection methods (in the "confirmation table") and a bar graph showing evidence used by the program to infer which of the sequences used to detect the recombination signal is the recombinant sequence. The most important bit of information displayed here is, however, the series of warnings that the program gives in capitalised red letters. These will indicate when RDP3 is reasonably unsure about some of the conclusions it has reached. The program will issue a warning if (a) one or both of the inferred breakpoint positions are probably inaccurate; (b) the wrong sequence may have been identified as the recombinant; (c) it is possible that an alignment error has generated a false positive signal; (d) there is only one sequence in the dataset resembling one of the recombinant's parental sequences; (e) only trace evidence of a recombination event is present within the currently specified sequence (*see* **Note 1**).

The automated output given by the program is nothing more than a preliminary hypothesis describing a small fraction of the recombination events that have occurred during the histories of sequences you have analysed. It is very important that you be aware that the program is fallible. The program's failures will be of four major types:

(1) Inaccurate identification of recombination breakpoint positions.

(2) Incorrect identification of parental sequences as recombinants.

(3) Incorrect inference that groups of identified recombinants are all descended from the same recombinant ancestor.

(4) Incorrect inferences that recombinants descended from the same ancestral recombinant contain evidence of unique recombination events.

Unfortunately there is not yet any automated tool in RDP3 that will enable you to definitively judge whether the results you obtain contain any of these errors. It is very likely that, unless the initial automated run of RDP3 indicates only a few recombinant sequences (<20% of the sequences in the dataset are recombinant), the program will have made some mistakes interpreting the patterns of recombination it has detected. The size and importance of the mistakes will scale with the number of unique recombination events the program detects. It is especially important to realise that mistakes early on in the analysis (such as in the first 10% of unique recombination events the program has characterised) will be more serious than those made in the end stages of the analysis. The

reason is that RDP3 identifies and characterises the easiest to detect recombination events first and leaves the interpretation of the least obvious recombination signals until last. Once, for example, a mistake has been made identifying which of the sequences is recombinant for a specific recombination event, the probability that the program will make additional mistakes of this type during the characterisation of all subsequent recombination events will be increased.

2.4. Navigating Through the Results

It is strongly recommended that you refine the recombination hypothesis the program provides and that you do this one recombination event at a time. As mistakes early on in the analysis are likely to be more serious than mistakes towards the end, you should start the refinement process with the recombination event that the program characterised first. All the unique recombination events detected by RDP3 are numbered in order from the first to the last that the program characterised.

You can navigate through the detected events using the computer keyboard. Left click on a background region of the schematic sequence display and then move between events in the order that the program detected them using the "Pg Up" and "Pg Dn" buttons.

For each event you move to, a graph of the clearest recombination signal used to detect the event is presented in the plot display and information relevant to the event is presented in the recombination information display (**Fig. 9.1**).

2.5. Checking the Accuracy of Breakpoint Identification

Graphs in the plot display (**Fig. 9.1**) are useful for checking the accuracy of recombination breakpoint estimation. Probably the most useful graph for this purpose is that generated by the MAXCHI method (*see* the plot display in **Fig. 9.3**). Whereas the positions of breakpoints are indicated by intersecting lines in SISCAN, RDP and BOOTSCAN plots, they are indicated by peaks in MAXCHI and CHIMAERA plots. If breakpoint positions indicated by different methods do not match, this indicates that there is a fair degree of uncertainty regarding the position of the breakpoints. Ranked in approximate order of breakpoint prediction accuracy (from best to worst) the different methods are as follows:

MAXCHI/CHIMAERA > 3SEQ > RDP > BOOTSCAN > SISCAN > GENECONV.

Another tool that can be used to determine the optimal locations of breakpoint pairs are MAXCHI and LARD matrices (**Fig. 9.3**). These graphically display all the probabilities of potential breakpoint pairs. To see a MAXCHI matrix press the matrices button on the top right hand panel (**Fig. 9.1**). Move the mouse pointer into the matrix window and press the left mouse button. Select the MAXCHI matrix option from the menu that appears

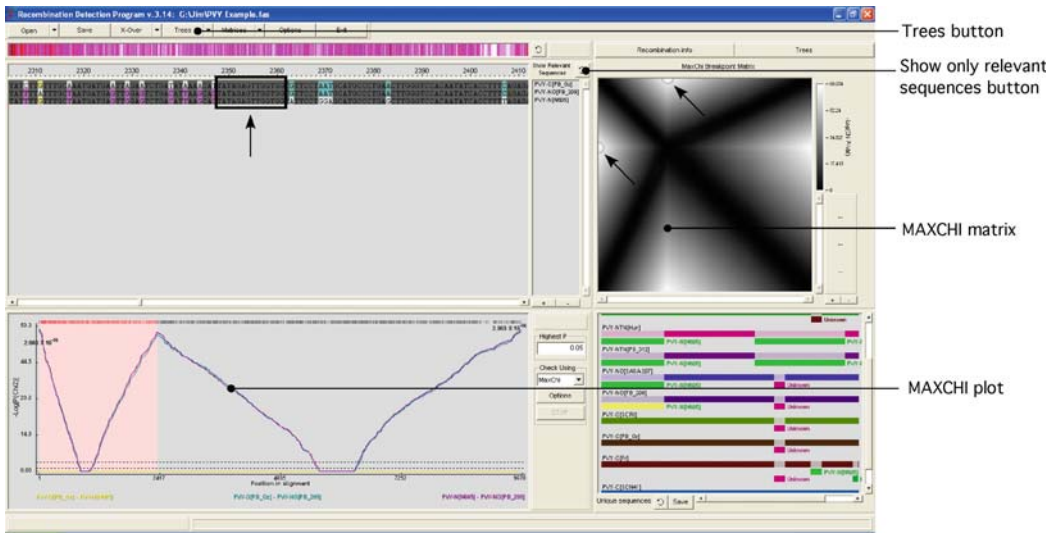


Fig. 9.3. Three tools for checking the accuracy of breakpoint prediction. The peaks of MAXCHI plots should coincide with the identified breakpoint positions. A MAXCHI matrix is similar to a maxchi plot but it expresses the maximum Chi square *P*-values of every possible pair of breakpoints. The "peaks" in the displayed plots are the white regions. *Arrows* indicate the pair of breakpoints identified by RDP3. Finally, the patterns of polymorphic sites in a sequence triplet can help indicate that the actual breakpoint position probably occurs somewhere within a range of identical sites (indicated by a *black box*). The site identified as the breakpoint position is indicated by a *vertical arrow*.

and the MAXCHI matrix will be displayed (**Fig. 9.3**). Interpretation of the matrix is relatively simple in that the most probable (although not necessarily correct) breakpoint pairs correspond with matrix cells that have the best associated *P*-values (use the colour key displayed beside the matrix to determine which colour corresponds with the best *P*-value).

If you would like to alter the position of one or both of the breakpoints, this can be done via the sequence alignment display window. You will need to go to the "show relevant sequences" version of this display. You get there by repeatedly pressing on the curled arrow in the top right hand corner of the sequence alignment display (**Fig. 9.3**) until the caption beside the arrow reads "show relevant sequences." Once there, you can get to the approximate region of sequence where you think the breakpoint should be by moving the mouse cursor to the corresponding point on the graph and pressing the left mouse button twice in quick succession. The "show relevant sequences" version of the alignment display will allow you to decide the best point to place the breakpoint as it colour codes variable nucleotide positions in the alignment according to which of the three sequences used to detect the recombination signal is most closely related. You ideally would like to place the breakpoint position in-between two variable nucleotides, one of which indicates a close relationship to one parent and

the other indicating a close relationship to the other. When you have decided on a position, point the mouse cursor at it and press the right mouse button. On the menu that appears some of the options involve placing breakpoints at this position. Some refer to "beginning/ending breakpoints" and others refer to "ancestral beginning/ending breakpoints." Placing an ancestral breakpoint will automatically adjust the breakpoint in all other sequences descended from the same ancestral recombinant as the currently selected sequence. Placing a breakpoint rather than placing an ancestral breakpoint will modify the breakpoint position only in the currently selected sequence.

2.6. Checking the Accuracy of Recombinant Sequence Identification

The next thing to consider is whether the program has correctly identified the recombinant. This can be very difficult to asses. The program uses a range of phylogenetic and genetic distance based tests to infer which sequence is the recombinant in a group of sequences containing a recombination signal. Very often different tests tell the program that different sequences are the recombinants and the program therefore uses a weighted consensus of these tests (*see* **Note 2**). The results of these tests are displayed, together with their weighted consensus, as a series of bar graphs in the "recombination information display" (**Fig. 9.1**). RDP3 will provide a warning if the tests do not clearly indicate which sequence is recombinant. You must take this warning seriously and determine for yourself whether one of the suggested parental sequences might not be the actual recombinant.

The best available tool in RDP3 for assessing whether the recombinant sequence has been correctly identified is to draw and compare two phylogenetic trees, one constructed from the portion of the alignment between the inferred breakpoints, and the other from the remainder of the alignment. The program automatically draws UPGMA trees for each of the two sections of the alignment whenever a particular recombination event is selected for more detailed analysis. You can see these trees side-by-side if you press the "Trees" button at the top of the screen (**Figs. 9.3** and **9.4**). If you would prefer a bootstrapped neighbour joining tree (built using PHYLIP; *10*), least squares tree (also built using PHYLIP) or maximum likelihood tree (built using PHYML; *11*), you can change the tree type that is displayed by (a) moving the mouse cursor over one of the trees, (b) pressing the right mouse button, (c) selecting the "change tree type" option offered at the bottom of the menu that appears, and then (d) choosing the preferred tree type from this sub-menu.

If you feel the program has not correctly identified the recombinant you should be very careful when deciding to change the choice that the program has made as there are many factors that might seriously complicate the identification of recombinant sequences using phylogenetic trees (*see* **Note 4**). Always remember

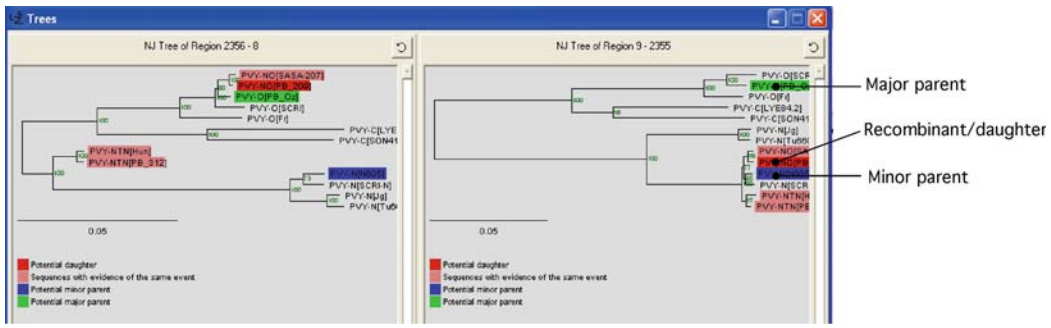


Fig. 9.4. Using phylogenetic trees to determine which sequence(s) is (are) recombinant. In this example RDP3 has inferred that four sequences (the PVY-N and PVY-NTN ones) are descended from a common recombinant ancestor with parental sequences resembling the PVY-N and PVY-O sequences. Note that the sequences do not form a monophyletic group in either tree. This may indicate either that RDP3 has "over-grouped" the sequences or that other recombination events elsewhere in the sequences may be obscuring the relationships between the four sequences.

that the program has used this phylogenetic approach along with a battery of other tests and has, for some (perhaps very good) reason, come up with a different solution. Also, if you think the program has made a mistake, be sure that you understand what the trees are telling you. For the sake of clarity RDP3 presents trees that are midpoint rooted, but you should realise that they are all actually un-rooted and therefore the direction of evolution may not be immediately clear. Your greatest chance of doing better than the program at identifying true recombinants is when the program has either identified two very close candidate recombinants (it will tell you which these are) or when you have obviously corrected a badly placed breakpoint position in the preceding step of the analysis. The reason for the latter is that the accuracy of the tests the program used to identify the recombinant sequence may have been compromised by the use of badly placed breakpoint positions.

If you would like to reassign the recombinant, you can do so by moving the mouse pointer over the graphical representation of the recombination event and pressing the right mouse button. A menu will appear and the options offered at the bottom of this will be to swap the recombinant and either the major or minor parental sequences. Select the appropriate option based on whether you think the recombinant has been misidentified as the major or the minor parent.

2.7. Checking How Well RDP3 Has Grouped Recombination Events

When you are interested in tracing the history of detectable recombination events in an alignment, it is desirable that RDP3 properly identify groups of sequences sharing evidence of the same unique recombination events. However, RDP3 might mistakenly group sequences that are descendents of different ancestral recombinants. The program could also mistakenly infer that two or more

unique recombination events are responsible for recombination patterns detected in sequences that are in fact all descended from the same recombinant. Although you might expect that the descendants of recombinant sequences should all have nearly identical mosaics and "move" together within phylogenetic trees constructed from different parts of an alignment, this is not always the case. For example, some sequences may contain only partial evidence of a particular recombination event because a second, newer recombination event overprinted part of the evidence from the older recombination event (*see* **Note 5**).

Another way of identifying sequences with evidence of the same ancestral recombination event is to generate and compare BOOTSCAN or RDP plots with different potential recombinant sequences while using the same parental sequences. To do this in RDP3, (a) make a list of sequences either that the program has indicated contain evidence of a similar recombination event, or that you have chosen based on, for example, their clustering with identified recombinants in phylogentic trees constructed from different parts of the alignment – in this case the list should include the sequences highlighted in pink in the phylogenetic trees (**Fig. 9.4**); (b) select the BOOTSCAN/RDP plot in the "check using" dropdown menu on the graph display (**Fig. 9.1**); (c) in one of the trees move the mouse pointer over one of the sequence names you have on your list, press the right mouse button and select the "Recheck plot with ... as daughter" option. If the two plots are very similar, it indicates that the sequences you selected contain similar recombination signals and are probably, therefore, different descendents of the same ancestral recombinant. If they are very different or only vaguely similar, there is a fair chance that the two recombination signals represent different unique events.

If you are convinced that RDP3 has either missed evidence that a group of sequences have all descended from a common ancestral recombinant, or incorrectly identified that a group of recombinant sequences have all descended from a common recombinant ancestor, the situation can be remedied via the tree display. On one of the trees, move the mouse pointer over one of the sequence names that have been incorrectly included or excluded as sharing evidence of the current recombination event. Press the right mouse button and the first option on the menu that appears will be to mark the sequence you have selected as either having or not having evidence of the current recombination event being analysed. You then select the appropriate option and the sequence will be added to or removed from the list of sequences containing evidence of that event.

2.8. Completing the Analysis

Having either refined or left the current recombination hypothesis unchanged, it is important that you tell the program that you are at least provisionally happy with the way the

current event has been characterised. You do this by moving the mouse pointer over the coloured rectangle representing the recombination even in the bottom right program panel (the rectangle should be flashing). Press the right mouse button and select the "Accept all similar" option near the middle of the menu that appears. A red border will be drawn around the coloured rectangle (and all others representing the same event) and this will tell the program that in all subsequent analyses you are happy with the characterisation of this event. If you are not happy with the way either you or the program has characterised the event you can opt to either leave the event unaccepted or you could select the "Reject all similar" option. When events are either accepted or rejected, RDP3 will skip these during your navigation through the remaining events that need to be checked.

If you did not modify breakpoint positions, change which sequence was recombinant and left the list of sequences sharing evidence of the same recombination event unchanged, then you can press the "Pg Dn" button and begin checking the next event. If, however, you modified the recombination hypothesis and accepted your modification, it is important that you reanalyse the data to account for your corrections. The reason for this is that the hypotheses proposed for all subsequent events analysed by the program were contingent on the unmodified version of the event you have re-characterised. Your changes, no matter how small, could influence the remainder of the analysis. To reanalyse the data taking your modifications into account, move the mouse pointer over the bottom right panel (the one with the coloured rectangles), press the right mouse button and select the "Re-Identify recombinant sequences for all unaccepted events" option from the menu that appears. Once the program has reanalysed the sequences, press the Pg Dn button and check the next event.

2.9. Saving Analysis Results

Once all events have been checked and you are either satisfied with your refined hypothesis, or you would simply like to continue analysing the data at a later stage, press the "Save" button on the menu bar at the top of the screen (**Fig. 9.1**). You will be given the opportunity to save the entire analysis in ".rdp" (RDP project file) format. If you would like a tabulated representation of the results, opt to save the data in ".csv" format. This format will allow you to open the results in spreadsheet programs such as Microsoft Excel. Note, however, that RDP3 will not be able to reload your analysis from a ".csv" file. You may also save any of the graphical outputs that RDP3 produces by right clicking on the appropriate program panel and selecting the "Save as..." or "Copy" options presented.

3. Examples

3.1. Producing a Preliminary Recombination Hypothesis

Load the example alignment file "PVY Example.fas" and press the "Options" button (**Fig. 9.1**). The example sequences we will be analysing are linear virus genomes, so under the "General settings" tab change the "Sequences are circular" setting to "Sequences are linear" (**Fig. 9.2**). Besides this change, we will use the default RDP3 settings for the example. Press the "OK" button at the bottom of the options form. Press the "X-Over" button (**Fig. 9.1**) and wait for the automated analysis to complete.

3.2. Navigating Through the Results

Press the left mouse button when the mouse pointer is on a background greyed area of the schematic sequence display (**Fig. 9.1**). This focuses the program on the display. Pressing either the "page down," "page up" or "space bar" keys on your computer keyboard will now allow you to navigate through the detected recombination events in an ordered fashion. Immediately after finishing the automated analysis pressing the page-down button will take you to the first recombination event that the program characterised. Pressing it again will take you to the second event, and so on. Pressing the page up button will take you to the previous event. Pressing the space bar will take you to the recombination event with the best associated *P*-value.

Starting with the first event (press the Pg Up or Pg Dn button until information on "event 1" is displayed in the recombination information panel) you will see a graph drawn on the "plot display" (**Fig. 9.1**). The exact information that is plotted in this graph will depend on the recombination method that yielded the best evidence of recombination for the recombination event at hand. In this case the plot is for the RDP method. The yellow, purple and green plotted lines each represent comparisons between different pairs of sequences in the sequence triplet used to detect the recombination event. The part of the graph between alignment positions 1 and 2,355 indicates that sequence identified as the recombinant, PVY-NO[PB_209], is most closely related to PVY-N(N605) in this region. Over the remainder of the sequence, between alignment positions 2,356 and 9,670, PVY-NO[PB_209] is most closely related to PVY-O(PB_Oz). The probability that this relationship between PVY-NO[PB_209] and the other two sequences could be explained by convergent evolution (i.e. without invoking a recombination hypothesis) is approximately 1 in 6×10^{216}. This is very good evidence of recombination. Look at the confirmation table in the recombination information display (**Fig. 9.1**). Note that all methods other than LARD and PHYLPRO *(12)* have also indicated that the shifting relationship between PVY-NO[PB_209] and the other

two sequences are good evidence of recombination. All associated *P*-values are smaller than 10^{-20}. The LARD and PHYLPRO methods were not used during the automated exploratory scan for recombination signals and this is the reason that they have no associated *P*-value. The large differences between the reported *P*-values for the other methods are largely due to the fact that the different methods look at slightly different signals in the alignment and have different approaches to approximating the probability that potential recombination signals are not caused by recombination.

Note that there is a warning (in red capitalised letters) in the recombination information display. Because you are analysing linear sequences, RDP3 is telling you that only the "Ending" (or 3') breakpoint is an actual recombination breakpoint (the other "breakpoint" listed is the beginning of the genome).

3.3. Checking the Accuracy of Breakpoint Identification

It is important that you check the accuracy with which RDP3 has identified the recombination breakpoint positions. Whereas the RDP method used to detect this recombination signal has intermediate accuracy, the MAXCHI method has higher accuracy when it comes to identifying breakpoint positions. To see a MAXCHI graph for event 1, press on the "check using" button on the right hand side of the plot display (**Fig. 9.1**). One of the options offered is to draw a MAXCHI plot. Select this option and see whether the peaks on any of the three lines plotted correspond with the borders of the detected recombinant region (shown in pink on the graph, **Fig. 9.3**) Look at graphs for some of the other methods. The boundaries of the recombinant region in pink should match positions in the RDP, BOOTSCAN, SISCAN and DISTANCE plots where two of the three plotted lines intersect. As with the MAXCHI plot, the boundaries of the pink region should match peaks in at least one of the lines in CHIMAERA, TOPAL *(13)*, PHYLPRO or LARD plots. For this recombination event all of the methods seem to indicate the recombination breakpoint at position 2,355 has been correctly identified (*see* vertical arrow in **Fig. 9.3**).

However, the actual breakpoint position might not be as obvious as you think. Note that in the recombination information display, four different sequences have been identified as descendents of the same event 1 recombinant (you can tell this by looking at the confirmation table in the recombination information display). Press the "Trees" button at the top of the screen (**Fig. 9.1**). Four of the sequences in the trees displayed (**Fig. 9.4**) are highlighted red or pink. These are the sequences carrying evidence of event 1. The sequence in red is the currently selected one of the four (it should be PVY-NO[PB_209]). Move the mouse pointer over PVY-NTN[Hun] and press the right mouse button. Select the "Go to PVY-NTN[Hun]" option. This will centre the schematic sequence display on PVY-NTN[Hun]. Move the mouse

pointer over the left most coloured rectangle representing the event 1 recombination signal in PVY-NTN[Hun]. Look at the recombination information display. Note that the "Ending" breakpoint position is identified here as 2,366 and not 2,355. This is a small but important difference. Press the "show relevant sequences button (**Fig. 9.3**) and use the scroll bar at the bottom of the sequence display to move to position 2,366. The colour coding of the nucleotides now corresponds with the colours of the lines in the plot below. You will see that at position 2,363, PVY-NTN[Hun] and PVY-N[N605] share an A nucleotide and the breakpoint is inferred to lie three nucleotides to the right of this point in PVY-NTN[Hun]. Now go back to the corresponding representation of event 1 in PVY-NO[PB_209] and left click on it. Look at the sequence display and you will see that at position 2,363 the polymorphic nucleotide of PVY-NO[PB_209] is shared with PVY-O(PB_Oz).

Clearly the breakpoint position should be somewhere in the region between 2,349 and 2,368, but its precise location is a mystery. Let us, just for the sake of this example, adjust the breakpoint position to position 2,359, near the middle of this region. To do this move the mouse pointer to alignment position 2,359 over the middle sequence in the sequence display and press the right button. One of the options that are offered is to "Place ancestral ending breakpoint here." Select this option and when you look at the representations of this event in the schematic sequence display you will see that they all report that the breakpoint position is at 2,359. If you had selected the "Place ending breakpoint here" option, the estimated breakpoint position would have only been changed in PVY-NO[PB_209].

3.4. Checking the Accuracy of Recombinant Sequence Identification

Look at the bar graphs in the recombination information display (**Fig. 9.1**). The first set of three bars indicate the consensus "Daughter scores" of PVY-NO[PB_209] (0.605), PVY-O[PB_Oz] (0.190), and PVY-N[N605] (0.204). These scores are the weighted consensus of a series of tests (each indicated by a set of three bars in the graph) to determine which sequence out of the three is the recombinant. In this case it is clear that PVY-NO[PB_209] is probably the recombinant. Treat these daughter scores as you would a bootstrap values in a tree – i.e. while their statistical meaning of the daughter scores are obscure, they are still an objective measure of which sequence is most probably the recombinant.

To check RDP3's decision that PVY-NO[PB_209] is the recombinant it would be useful to look at phylogenetic trees. Bring up the side-by-side tree display by pressing the "Trees" button (**Fig. 9.3**). Right click on an empty part of one of the two windows. The menu that appears has an option to "Change tree type." Select this and then select the neighbour joining tree

type. Now change the tree in the other window to neighbour joining too. You should see the same trees as in **Fig. 9.3**. For now focus on the sequences in the tree highlighted in red (PVY-NO[PB_209]), green (PVY-O[PB_Oz]) and blue (PVY-N[N605]) and ignore those highlighted in pink. Comparing the locations of these three sequences in the trees should indicate that the sequence highlighted in red has moved from the PVYC/PVY-O clade of the tree to the PVY-N clade (*see* **Note 3**). It seems that the program was correct to identify this sequence as the recombinant. You therefore do not need to change anything.

However, for the sake of this example, right click on the flashing rectangle in the schematic sequence display and select the "Swap daughter (PVY-NO[PB_209]) and minor parent (PVY-N[N605])" option. See how the flashing rectangle is "moved" to PVY-N[N605]. Look at the tree and see how interpretation of this recombination event has changed. Now swap the event back to PVY-NO[PB_209].

3.5. Checking How Well RDP3 Has Grouped Recombination Events

There appears to be some evidence that the current recombination event, "event 1," may have occurred in the common ancestor of four sequences in the dataset – those sequences currently highlighted in pink/red in the trees and whose names are prefixed with PVY-NTN and PVY-NO.

In our example dataset it is apparent that the four identified recombinant sequences do not all move together between the phylogenetic trees. However, both the two PVY-NO sequences and the two PVY-NTN sequences do move together. In the left tree in **Fig. 9.4** notice how the PVY-NTN and PVY-NO sequences are both clearly on separate well supported branches. If you look at the sequences PVY-NTN[Hun] and PVY-NTN[PB_312] in the schematic sequence display you will immediately see the probable cause for their being located on a separate branch to the PVY-NO sequences in the left tree. RDP3 has identified another large recombination event in both of the PVY-NTN sequences with breakpoints at alignment positions 5,777 and 9,141.

Recheck the plot with PVY-NTN[Hun] as the daughter sequence. This will compare the plots of the currently selected sequence (in this case PVY-NO[PB-209]) with that of PVY-NTN[Hun]. The results of the comparison are displayed graphically in the form of a coloured line above the plot in the graph display (**Fig. 9.5**). Blue in this plot indicates regions of sequence where recombination signals are very similar and dark red, which occurs between alignment positions 5,777 and 9,141 (the other part of PVY-NTN[Hun] that RDP3 has identified as having a recombinant origin), indicates regions of sequence where recombination signals are completely different. If the portion of sequence spanning one or both recombination breakpoints

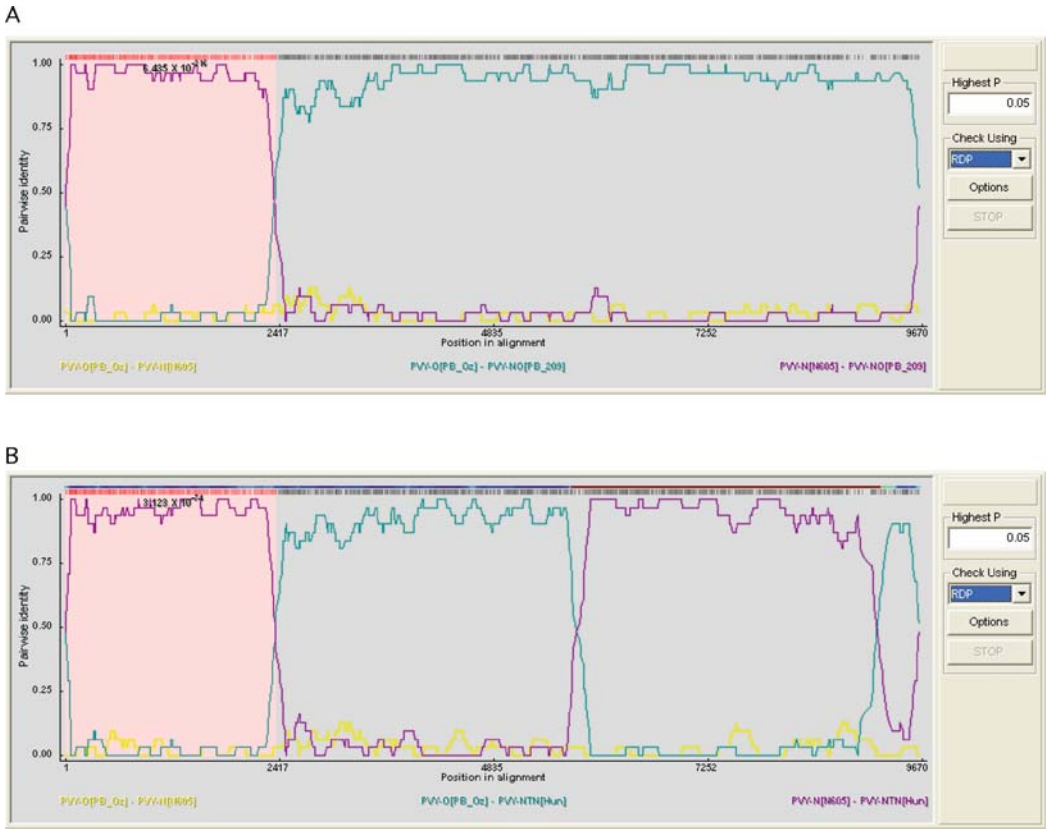


Fig. 9.5. Comparing recombination signals to determine whether two recombinants are descended from a common recombinant ancestor. (**A**) A RDP method plot for Event number 1 in the example dataset. (**B**) A similar RDP plot as in **A** but with PVY-NTN[Hun] replacing PVY-NO[PB_209] in the scanned sequence triplet. The *coloured line* above the plot is a graphical representation of how closely the plot in **B** resembles that in **A**. Note that across the recombination breakpoints the two plots are nearly identical (the *blue* color in the *bar* also indicates this) indicating that both PVY-NTN[Hun] replacing PVY-NO[PB_209] probably descended from the same recombinant ancestor. However, in **B** both the plot and deep *red* colour in the part of the *bar* above the plot indicate very clearly that PVY-NTN[Hun] carries evidence of a second major recombination event not shared with PVY-NO[PB_209].

corresponds with a blue signal, as it does in this case, then the pattern of sites shared by the sequences being compared and their supposed parental sequences are very similar and are possibly derived at approximately the same time from the same or similar sources.

However, for the sake of this example, let us pretend that the recombination signals in the PVY-NTN sequences are not derived from the same ancestral recombinant as that of the signals in the two PVY-NO sequences. To exclude the PVY-NTN sequences from event 1, go to the side-by-side tree display and right click on PVY-NTN[Hun]. Choose the option to "Mark PVY-NTN[Hun] as not having evidence of this event." Now do the same for PVY-NTN[PB_312].

If you would like to re-include the signals within PVY-NTN[Hun] and PVY-NTN[PB_312] as being descended from the same unique recombination event as those detected in the PVY-NO sequences, then right click on the sequence names in the tree and select the "Mark <sequence name> as having evidence of this event" option.

3.6. Completing the Analysis

Because you have changed the way that RDP3 has interpreted event 1 you need to let the program reformulate its opinion on what has happened in all the other recombination events it has detected. First, however, it is important to tell RDP3 that you are happy with the current interpretation of event 1. To do this, go to the flashing rectangle in the schematic sequence display and right click on it. Select the "Accept all similar" option and you will notice a series of red borders have been drawn around the rectangles representing the event 1 signal in the PVY-NO sequences (and in the PVY-NTN sequences if you added them back to event 1). Now right click anywhere in the schematic sequence display and select the "Re-identify recombinant sequences for all unaccepted events" option.

When RDP3 has finished reanalysing the remaining recombination signals press the "Pg Dn" button on your keyboard and you can start evaluating event 2. Continue until you reach the end of the analysis. events 3 and 4 are examples of apparently reciprocal recombination (PVY-N[SCRI-N] and PVY-O[Fr]). If you are interested in seeing the effect of the "Disentangle overlapping events" setting (**Fig. 9.2**), see what happens with this event when the dataset is analysed with and without the setting on. event 6 is an example of a very marginal event only detectable with the MAXCHI and CHIMAERA methods. Check it with the other methods to get some idea of how these methods perform when recombination signals are weak.

4. Notes

1. When RDP3 attempts to infer the number of unique detectable recombination events in an alignment, it detects a recombination signal and tries to determine which other sequences in the alignment carry similar recombination signals. Often sequences carrying similar signals will be identified but the signal in these sequences is not strong enough to generate a statistically significant *P*-value. These signals are referred to as "trace" signals and are listed as such in both the "recombination information" part of the RDP3 display and the

phylogenetic trees that the program constructs to help you decide which sequences are recombinant and which are relatives of parental sequences.

2. There is also no guarantee that the relative weighting of the tests is accurate. Tests using recombinant HIV sequences (for which parental viruses that have been epidemiologically separated until recently) have been used to weight the different tests. However, what may be a reasonably accurate weighting for HIV might not be good for your data. Also, even in tests with HIV the program only has an approximately 90% success rate when it comes to correctly identifying the recombinant sequence.
3. RDP3 allows you to mark particular sequences in the trees that it displays. This can be very useful for tracking the "movement" of particular sequences between the clades of different trees. You can mark a sequence by moving the mouse pointer over the name of the sequence in the tree and pressing the left mouse button. The same sequence is then marked in all of the trees. You can clear markings or automatically colour sequence names (so that they are the same colours as those displayed in the bottom right panel for graphical representations of the sequences) by selecting the appropriate option on the menu that appears whenever you press the right mouse button with the mouse pointer over one of the tree displays.
4. RDP3 scans every sequence triplet in an alignment to identify recombination signals and, given more than a single detectable recombination event in a dataset, some triplets will contain two or three recombinant sequences. While this can be a particularly serious problem if two or three of the sequences in the sequence triplet have breakpoints close to one another, it can also be a problem if their breakpoints are in completely different parts of the sequence. If the breakpoints are close together, the program will possibly infer a recombination event with two breakpoints each of which comes from a different recombinant sequence in the triplet. This will seriously influence how effectively trees can be used to judge which of the sequences is "most" recombinant (they are all recombinant but none have the inferred breakpoint pair). Even when the correct breakpoint pairs are identified, when two of the sequences in a triplet are recombinant it will often lead to the non-recombinant parent being identified as the recombinant.
5. If recombination has occurred frequently within the history of a sample of sequences then there is a good chance that evidence will exist in the alignment of older recombination events being overprinted by newer ones. When recombination

events are partially overprinted in some sequences descended from an ancestral recombinant but not in others there is a chance that the program will not group the recombination signals properly and it may infer that two recombination events have occurred instead of one. Although the inference of two recombination events is technically correct in such cases, the program will incorrectly identify which sequence exchanges took place. It can be very difficult to interpret the phylogenetic signals caused by partially or completely overprinted events.

References

1. Martin, D. P., Williamson, C., and Posada, D. (2005) RDP2: recombination detection and analysis from sequence alignments. *Bioinformatics* **21**, 260–2.
2. Martin, D., and Rybicki, E. (2000) RDP: detection of recombination amongst aligned sequences. *Bioinformatics* **16**, 562–3.
3. Padidam, M., Sawyer, S., and Fauquet, C. M. (1999) Possible emergence of new geminiviruses by frequent recombination. *Virology* **265**, 218–25.
4. Maynard-Smith, J. (1992) Analyzing the mosaic structure of genes. *J Mol Evol* **34**, 126–9.
5. Posada, D., and Crandall, K. A. (2001) Evaluation of methods for detecting recombination from DNA sequences: Computer simulations. *Proc Natl Acad Sci USA* **98**, 13757–62.
6. Martin, D. P., Posada, D., Crandall, K. A., and Williamson, C. (2005) A modified bootscan algorithm for automated identification of recombinant sequences and recombination breakpoints. *AIDS Res Hum Retroviruses* **21**, 98–102.
7. Boni M. F., Posada, D., and Feldman, M. W. (2007) An exact nonparametric method for inferring mosaic structure in sequence triplets. *Genetics* **176**, 1035–47.
8. Gibbs, M. J., Armstrong, J. S., and Gibbs, A. J. (2000) Sister-Scanning: a Monte Carlo procedure for assessing signals in recombinant sequences. *Bioinformatics* **16**, 573–82.
9. Holmes, E. C., Worobey, M., and Rambaut, A. (1999) Phylogenetic evidence for recombination in dengue virus. *Mol Biol Evol* **16**, 405–9.
10. Felsenstein, J. (1989) PHYLIP – Phylogeny Inference Package (Version 3.2). *Cladistics* **5**, 164–6.
11. Guindon, S., and Gascuel, O. (2003) A simple, fast, and accurate algorithm to estimate large phylogenies by maximum likelihood. *Syst Biol* **52**, 696–704.
12. Weiller, G. F. (1998) Phylogenetic profiles: a graphical method for detecting genetic recombinations in homologous sequences. *Mol Biol Evol* **15**, 326–35.
13. McGuire, G., and Wright, F. (2000) TOPAL 2.0: improved detection of mosaic sequences within multiple alignments. *Bioinformatics* **16**, 130–4.

Chapter 10

CodonExplorer: An Interactive Online Database for the Analysis of Codon Usage and Sequence Composition

Jesse Zaneveld*, Micah Hamady*, Noboru Sueoka, and Rob Knight

Abstract

The analysis of DNA composition and codon usage reveals many factors that influence the evolution of genes and genomes. In this chapter, we show how to use CodonExplorer, a web tool and interactive database that contains millions of genes, to better understand the principles governing evolution at the single gene and whole-genome level. We present principles and practical procedures for using analyses of GC content and codon usage frequency to identify highly expressed or horizontally transferred genes and to study the relative contribution of different types of mutation to gene and genome composition. CodonExplorer's combination of a user-friendly web interface and a comprehensive genomic database makes these diverse analyses fast and straightforward to perform. CodonExplorer is thus a powerful tool that facilitates and automates a wide range of compositional analyses.

Key words: Database, sequence composition, codon usage, codon adaptation index, gene transfer, genomic islands, web interface.

1. Introduction

Different organisms use codons for the same amino acid at different frequencies and this non-randomness can be exploited for a variety of analyses of both theoretical and practical importance. Soon after the genetic code was first elucidated *(1, 2)* it was clear that it was "degenerate" – not in the sense of decaying from a more elaborate form, but in the physicist's sense of having multiple

* These authors contributed equally to this work

David Posada (ed.), *Bioinformatics for DNA Sequence Analysis,* Methods in Molecular Biology 537

DOI 10.1007/978-1-59745-251-9_10

equivalent states. In this case, because there are 64 codons but only 20 amino acids that are universal to all life, some amino acids must be encoded by more than one codon (exceptions to this rule, such as selenocysteine and pyrrolysine, are important but lineage-specific). We might expect these equivalent codons to be used at random, but early studies of nucleotide frequencies *(3)* suggested that different organisms have extreme biases toward certain synonyms. The availability of the first DNA sequences for protein-coding genes *(4–6)* confirmed that codon usage is in fact highly non-random and differs dramatically among organisms.

Why should different organisms use codons that have the same meaning with such different frequencies? An innovative analysis of highly-expressed genes by Ikemura et al. showed that genes that are known to be required at high levels, such as ribosomal proteins, have substantially different codon usage from the average gene in the same genome *(7)*. This suggested a strong effect of translation bias on codon usage because the tRNA pool in a given organism would recognize different anticodons with different efficiencies. For example, the GNN anticodon is often used to recognize either CNN or UNN at the first position in so-called "wobble pairing" *(8, 9)*, but the efficiency of these two pairings can differ substantially. Consequently, highly-expressed genes would be expected to be composed of codons that match the tRNA anticodons that are expressed at the highest levels and/or have the highest copy numbers; this effect has been confirmed in bacteria *(10, 11)*, yeast *(12)*, Drosophila *(13)*, and many other species. Thus, selection for translational efficiency has been shown to be one important influence on codon usage bias.

Many different measures of the codon usage bias in a given gene have been developed. One of the most popular is Sharp and Li's codon adaptation index (CAI) *(14)*, which indicates how closely the codon usage of each gene matches the codon usage of a set of known highly expressed genes. The relationship between codon usage and expression has proven useful for optimizing the expression of synthetic genes transferred into a host (reviewed in *(15)*; *see also* **Note 1** on the "Synthetic Gene Designer" website).

Selection for translational efficiency is not, however, the only influence on codon usage; the overall composition of the genome also has a strong effect on which codons are chosen. For example, Muto and Osawa were able to show surprisingly strong, linear trends relating the GC content of the whole genome to the GC content at the first, second, and third codon positions in a diverse assortment of 11 organisms *(16)*. Knight et al. confirmed that these trends held true in hundreds of different organisms across all three domains of life and demonstrated that the codon usage of a given organism could be predicted with considerable accuracy simply from a knowledge of these overall trends and the GC content of the individual organisms *(17)*. Consequently, both

selection and mutation play a large role in structuring the codon usage of genes and genomes. In fact, multivariate analyses of codon usages typically identify expression and GC content of individual genes as the principal factors structuring composition within a genome *(18)*.

Because codon usage is strongly influenced by mutational processes, codon usage can provide insight into the different mutational processes operating in different genomes *(19–21)*, or in different parts of the same genome where regions of compositional heterogeneity such as isochores exist *(22, 23)*. The ratio of different kinds of codons can provide insights into whether deamination or oxidation contributes more to the pattern of codon usage in a specific organism through techniques such as the fingerprint plot and the PR2 bias plot *(24–26)*.

Finally, codon usage and other compositional information can be used to detect horizontal gene transfer (HGT), the movement of genes between different genomes *(27)*. Because different organisms differ in the composition of their genes, HGT can be detected, if sufficiently recent *(28)*, by looking for genes of unusual composition *(29, 30)*. However, some caution must be exercised in this approach (*See also* **Note 2**), since highly expressed genes can also show codon bias, and gradients in composition can appear along a genome due to replication-coupled biases *(31, 32)*.

2. Program Usage

CodonExplorer (available at http://bmf.colorado.edu/codonexplorer/) provides a platform for conducting many diverse analyses of codon usage and nucleotide composition in sequenced genes or genomes. One feature of CodonExplorer is the ease with which millions of sequenced genes and hundreds of genomes can be searched, grouped, and analyzed. By pre-computing statistics from whole-genome databases, using hundreds of hours of CPU time, CodonExplorer is able to provide graphical summaries of vast, complex datasets very rapidly.

2.1. Gene Selection

The first step in performing sequence composition analysis using CodonExplorer is to gather the sequences on which the analysis will be performed. CodonExplorer incorporates multiple convenient methods for selecting nucleotide sequence data.

The first step when using CodonExplorer is to select a set of genes to analyze. There are several different search methods, including by KEGG gene IDs (*see* **Note 3**), KEGG orthology (KO) groups, or by enzyme commission (EC) ID (*see* **Note 4**). All genes in one or more genomes (specified by KEGG genome ID or using

the NCBI taxonomy) can be conveniently gathered, and a gene screen allows selection of highly expressed genes or putatively horizontally transferred genes within a genome. Finally, many genomic and pathogenicity islands (regions of putatively transferred genes) from the literature have been included and can be easily selected.

Once sets of genes have been selected, they can be sorted, grouped, and edited manually to remove undesired sequences or further refine the set. The KEGG gene IDs for the final set of sequences can be downloaded to allow easy reproduction of the analysis.

2.1.1. Selecting Genes by Gene or Genome

CodonExplorer allows genes to be selected with a list of KEGG genomes, KEGG gene IDs, KEGG gene names, and/or NCBI taxonomy names (**Fig. 10.1**) (*see* **Note 5**).

Gather genes by KEGG genome: To gather all genes in one or more genomes, a list of KEGG genome IDs (*see* **Note 6**) may be entered and the genes from each genome will be grouped and sorted according to the selected option (see grouping and sorting data, below).

Gather genes by NCBI taxon name: To gather all genes in an NCBI taxon, enter the NCBI taxon name in the field labeled "Enter NCBI taxon name(s)" and press the "Search for Genes" button (*see* **Notes 6**, **7**).

Gather genes by KEGG ID or gene name: To gather a list of genes by KEGG ID or gene name, enter the IDs or gene names in the appropriate fields. If search terms are entered in multiple fields, only genes in the intersection of terms are returned (i.e., the terms are combined with an "AND" operator).

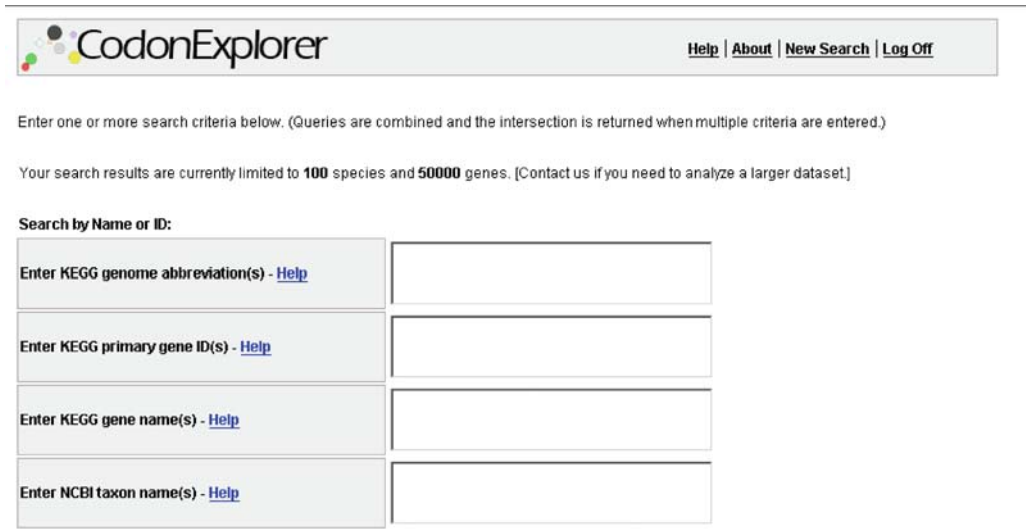


Fig. 10.1. Selecting genes by name or ID.

Example: *Searching for KEGG genome "STM" and KEGG gene name "rpsu" returns only the rpsU gene from* Salmonella enterica*serovar Typhimurium LT2; searching for KEGG genome "STM" will return every gene from* Salmonella enterica *serovar Typhimurium LT2.*

2.1.2. Selecting Genes by Ortholog, EC, or Gene Region

When a set of a precise set of gene IDs is not known in advance, CodonExplorer provides several convenient ways to select groups of genes sharing particular properties (**Fig. 10.2**).

Selecting Genes by KEGG Orthology (KO) Group: It is often useful to analyze related genes across several genomes. Codon Explorer facilitates this type of analysis by allowing users to select members of a desired KO group. (*See also* **Notes 8**, **9**.)

Example: To select homologs of recA, enter "K03553" into the KEGG ortholog or KEGG EC ID box, select "KEGG Ortholog" from the "Select type of ID" pull-down menu, and press the "Search for Genes" button at the bottom of the page.

Selecting Genes by Enzymatic Activity: CodonExplorer allows for users to search for genes encoding enzymes that share a common activity, as defined by their KEGG EC ID. To select a group by KEGG EC ID, enter the ID or list of IDs into the "KEGG ortholog or KEGG EC ID" and select "KEGG EC" from the "Select type of ID(s)" menu (*see* **Note 10**).

Example: *To search for α-amylases (EC3.2.1.1), enter EC3.2.1.1 into the KEGG ortholog field and select "KEGG Ortholog" from the "Select type of ID(s)" pull-down menu. Then press the "Search for Genes" button at the bottom of the page.*

Selecting Genes by Genomic Region: CodonExplorer allows users to select a range of genes within a genome. To do so, enter the KEGG gene ID for the first and last genes within the region in the "Enter KEGG gene range" field and select "Forward Only," "Reverse Only," or "Both Directions" from the "Gene region direction" pull-down menu (*see* **Note 11**).

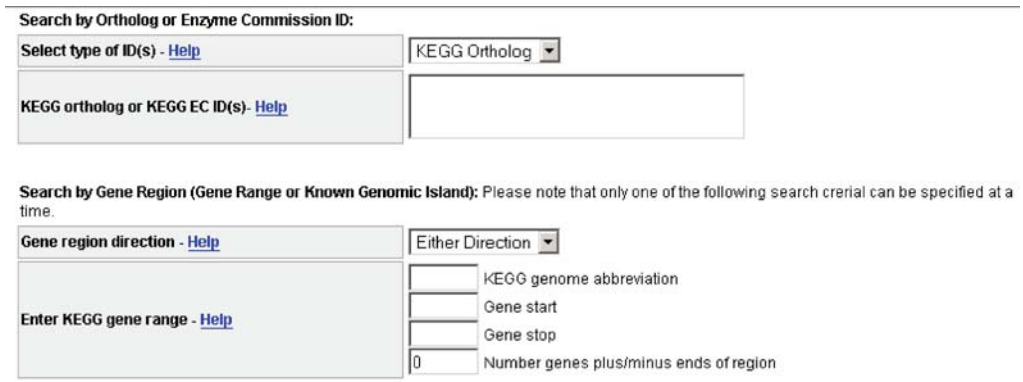


Fig. 10.2. Selecting genes by KEGG orthology (KO) group, enzyme commission (EC) ID, or gene region.

Example: *To select genes in either orientation between STM1379 and STM1422 in the* Salmonella enterica *serovar Typhimurium genome (i.e., the Salmonella Pathogenicity Island-2 region), enter "STM1379" in the "Gene Start" field, "STM 1422" in the "Gene End" field, "STM" in the KEGG genome ID field, select "Either direction" in the "Gene region direction" pull-down menu, and press the "Search for Genes" button at the bottom of the page.*

2.1.3. Selecting Genes in Genomic Islands or by Gene Screen

Selecting Genes in Pre-Annotated Genomic Islands: Many regions of genes present in a particular genome but absent in closely related lineages have been described in the literature *(33, 34)*. Frequently these regions, often called "genomic islands," have unusual dinucleotide compositions or contain mobility genes such as integrases or transposases *(34)*. Thus, it is suggested that genomic islands are the product of HGT. A subset of these genomic islands called "pathogenicity islands" also contains pathogenicity determinants, making them particularly interesting.

CodonExplorer allows users to select genomic islands that have been described in the literature by selecting a genomic island from the "Select genomic island" pull-down menu (**Fig. 10.3**). The genomic islands used in Hsiao et al.'s 2005 analysis *(34)* are included, as are many other islands collected manually from the literature (*see also* **Note 12**).

Selecting Highly Expressed or Putatively Transferred Genes: For codon usage or other compositional property analyses, it is often useful to explore variation in highly expressed genes (e.g., those encoding ribosomal proteins) or genes that may have entered the genome by HGT.

To select such a set of genes in CodonExplorer, enter the KEGG genome name in the "KEGG genome" in the gene screen section, then select "Ribosomal + highly expressed KO groups," "Putative Aliens," or "Not putative aliens" from the pull-down menu (**Fig. 10.3**). The putative and aliens and non-alien sets are defined using a Markov Model method described by Nakamura et al. *(30)*. However, like any compositional method, the Nakamura method should not be considered definitive evidence of transfer. See also the discussion below about the benefits and limitations of compositional methods for detecting gene transfer.

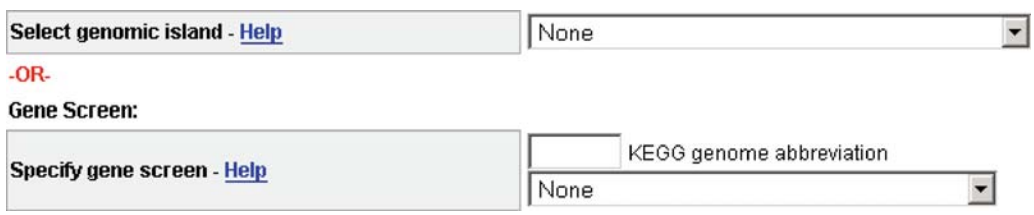


Fig. 10.3. Genomic islands and gene screen.

2.2. Output Options for Sorting and Grouping Selected Genes

By default, genes selected for analysis in CodonExplorer are grouped together by genome and sorted in descending order of the number of selected genes in each genome. However, it is sometimes useful to group and sort genes in different ways.

Grouping selected genes: CodonExplorer allows genes to be grouped by KEGG genome name, KEGG ortholog ID, KEGG EC ID, or (when NCBI taxon names were entered in the search) NCBI Taxonomy name (**Fig. 10.4**). To change how genes are grouped, enter your search terms as described above, then select the appropriate option from the "Group matching genes by" pull-down menu before pressing the "Search for Genes" button.

Sorting selected genes: CodonExplorer also allows groups of genes to be sorted by the number of genes in the group, the group ID, or the group description (**Fig. 10.4**). Select the desired option from the "Sort gene groups by" pull-down menu under the "Output options" heading.

2.3. Search Results – Viewing, Editing, and Saving Sets of Genes

After searching for genes on which to perform an analysis, determining how those genes should be sorted and grouped, and pressing the "Search for Genes" button at the bottom of the page, the genes will be displayed at the top of the page (**Fig. 10.5**) and sorted and grouped according to the selected options (*see* **Note 13**). From this screen it is possible to view, edit, or save gene groups.

View genes: Each gene group can be viewed by pressing the "View" link to the right of the group description. This will bring up a window listing the genome, KEGG gene ID, KEGG gene

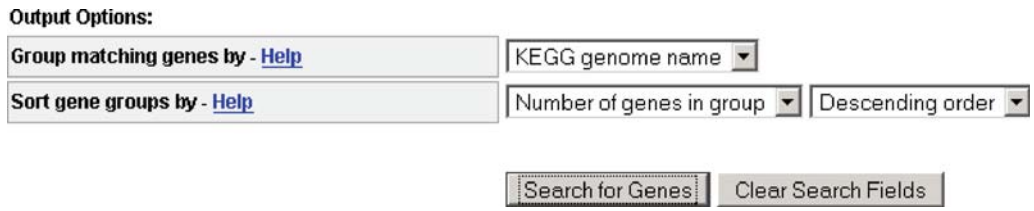


Fig. 10.4. Output options.

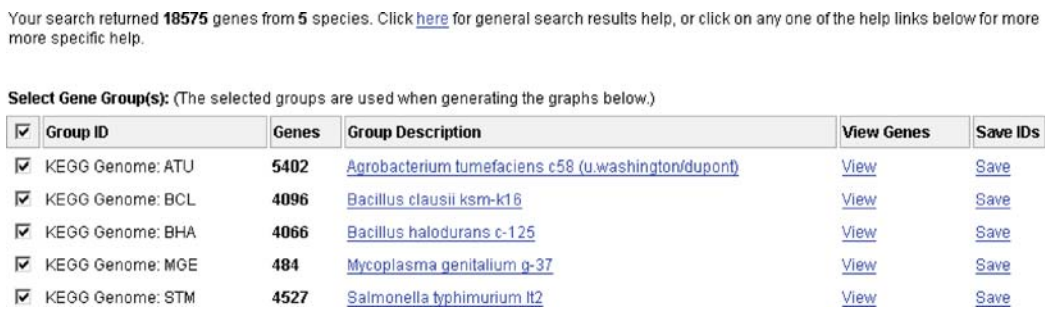
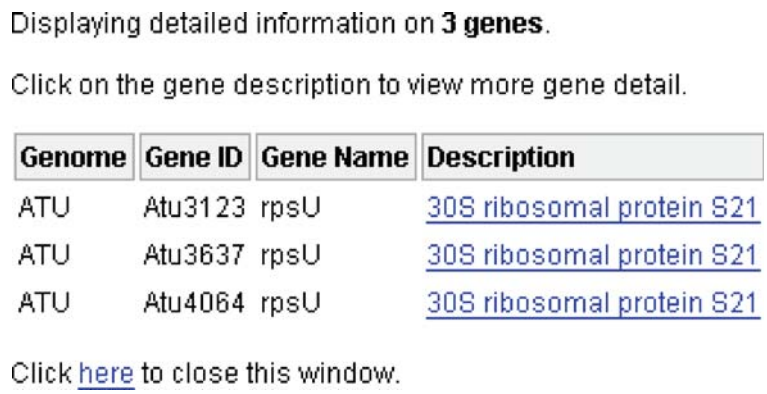


Fig. 10.5. Search results: all genes in genomes "ATU, MGE, STM, BHA, BCL."


Displaying detailed information on **3 genes**.

Click on the gene description to view more gene detail.

Genome	Gene ID	Gene Name	Description
ATU	Atu3123	rpsU	30S ribosomal protein S21
ATU	Atu3637	rpsU	30S ribosomal protein S21
ATU	Atu4064	rpsU	30S ribosomal protein S21

Click here to close this window.

Fig. 10.6. "View Genes" for rpsU group from ATU are shown.

name, and a description for each gene (**Fig. 10.6**). Clicking the "description" link will bring up the KEGG description page for that gene.

Include or exclude gene groups from analyses: To include or exclude one or more gene groups from the analyses, check or uncheck the check box to the left of each gene group's 'Group ID'.

Save genes: The KEGG primary gene IDs for genes in each group can be saved locally as a text file by clicking the "Save Genes" link next to that group.

2.4. Search Results – Selecting Analyses

Once one or more groups of genes have been selected, Codon Explorer allows many compositional analyses to be performed easily. In general, one or more analyses are selected, after which pressing the "Generate selected graph(s)" will cause a new window to appear displaying the desired plots. There are dozens of different analyses available, including comparisons of GC content at different codon positions, measures of codon usage bias, amino acid usage, and a Markov model method for HGT detection (**Fig. 10.7**). Custom scatter plots can also be constructed plotting many of these properties against one another, or against other properties such as length or hydrophobicity. Finally, a Monte Carlo method allows determination of the statistical significance of apparent differences between the properties of selected groups of genes and the genomes that contain them. For ease of use, we have classified these diverse approaches into the broad categories of "Codon Usage," "GC Content," and "Custom Graphs."

2.4.1. Codon Usage Analyses

Codon Fingerprint Plots: Fingerprint plots were introduced by Sueoka *(20)* to measure mutational biases within and between genomes. The *y*-axis represents the frequency of A at position 3 relative to the frequency of A and T at that position ($A_3/(A_3+T_3)$), while the *x*-axis represents $G_3/(G_3 + C_3)$. Each circle represents a different amino acid, with the radius of the circle

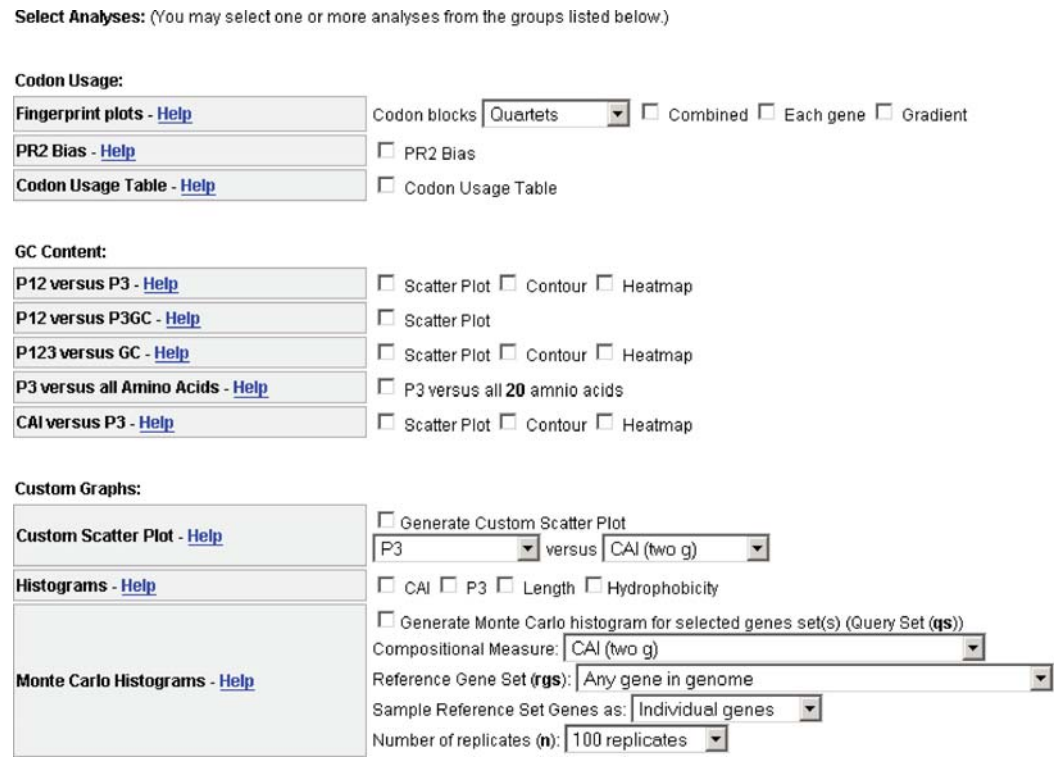


Fig. 10.7. Selecting analyses in CodonExplorer.

proportional to the relative frequency of that amino acid *(19)*. There are several options when generating fingerprint plots. The ***Quartets*** option generates a codon fingerprint plot using the four codon blocks (e.g., the CCN block coding for Proline, where N stands for any of the four nucleotides A, C, G, or T). These blocks are Leucine4, Valine, Alanine, Threonine, Proline, Serine4, Glycine, and Arginine4 (the 4 refers to the four-codon block of an amino acid that has six codons in total). The ***Split Quartets*** *option* generates a codon fingerprint plot using each of the six split codon blocks (e.g., the GAN block coding for Aspartic or Glutamic acid). These blocks include Phenylalanine/Leucine2, Isoleucine/Methionine, Histidine/Glutamine, Asparagine/Lysine, Aspartic Acid/Glutamic Acid, and Serine/Arginine2. The Tyrosine/Stop and Cysteine/Tryptophan/Stop partial blocks are not included on this plot.

Three graph types are available for codon fingerprint blocks in CodonExplorer. First, the ***Combined*** option plots all genes in the selected gene groups into a single fingerprint plot. This is useful when for summarizing large data sets, such as whole genomes, DNA strands within a genome, or genomic islands. Second, the ***Each Gene*** option allows for a finer grained analysis by plotting each gene within the set of genes selected for analysis separately

(*but see* **Note 14**). Finally, the ***Gradient*** option plot displays a combined fingerprint plot for genes within each 10% interval of GC content. Thus, depending on the number of genes selected, and the range of GC content represented by those genes, only some of the ten possible ranges of GC content will be displayed. An example of constructing and interpreting a gradient codon fingerprint plot is given in **Section 3**.

CAI versus P_3 Plots: As discussed above, GC content has an important influence on codon usage bias (Guy and Sueoka, manuscript in preparation). This influence can be illustrated by plotting CAI values against P_3 GC content (see the examples section for some plots of CAI vs. P3 in organisms with very different genomic GC contents).

PR2 Bias Plots: Chargaff's second parity rule (PR2) states that within each DNA strand, the frequency of A ≈ T and the frequency of G ≈ C *(35)*. Parity Rule bias plots were developed to examine deviations from PR2 *(26)*. Selecting the PR2 bias plot option generates a set of PR2 bias plots for the selected gene set.

2.4.2. GC Content Plots

GC content at different codon positions is used to study mutation and selection pressures. Because GC content varies dramatically between but not within bacterial lineages *(36)*, unusual GC content may indicate transfer. Thus unusual GC content has often been used as an indicator of gene transfer, although there are important limitations to this approach (see the example on HGT detection below). CodonExplorer allows the generation of several graphs that allow users to explore the GC content of genes and genomes.

P_{12} versus P_3 GC: Useful for contrasting the effects of mutation and selection for amino acid usage (P_1 and P_2) versus codon usage (P_3). The *x*-axis represents the GC content at the third position within each codon in the selected group of genes, and the *y*-axis represents the GC content at either the first or second position in each codon in the selected group of genes. (P_1 and P_2 GC contents are plotted as separate series.)

P_{12} versus P_3: Plots the combined GC for nucleotides at the first or second position in each gene against the GC content for nucleotides at the third position in that gene. The dotted diagonal line represents the expectation if all positions were under identical mutation and selection pressure.

P_{123} versus GC: Plots the first, second, and third position GC content (as separate series) against the overall GC content of the gene. Thus each point represents the P_1,P_2, or P_3 GC content of a gene, versus the total GC content of that gene.

P_3 versus All Amino Acids Plots: This plot allows the user to view a plot of the amino acid content of each gene within the group of selected genes against the P_3 GC content of that gene. It should be noted that since this approach divides the data contained

in each gene into 20 parts (by recording P_3 information for usage of each amino acid for each gene), these plots may be subject to large amounts of noise when examining small datasets. (*See* **Note 14** for a similar caution regarding gradient fingerprint plots).

2.4.3. Custom Graphs

Custom scatter plot: In addition to allowing the generation of many predefined plots, CodonExplorer also allows users to construct custom scatter plots using a variety of compositional statistics.

Values that can be plotted against one another on either axis include the CAI (as well as several proposed variations on it), the third position GC content (P_3), gene length, predicted peptide hydrophobicity, and the horizontal transfer index of Nakamura et al. *(30)* (**Table 10.1**).

Histograms: Histograms can be constructed of CAI values, GC content at the third codon position (P_3), gene length, or peptide hydrophobicity.

Codon Usage Table: Often it is useful to report a table of codon usages for a single gene or a collection of genes, e.g., to identify frequently used codons *(37)*. This can be done in Codon Explorer by selecting the "Codon Usage Table" option. The output is a table (**Fig. 10.8**) that lists the frequency of each codon per thousand codons in the dataset, followed by the absolute number of occurrences of the codon in the selected set of genes.

The total number of genes and codons in the data set are listed at the top of the table, and the overall GC content and the average GC content at positions 1, 2, and 3 are listed at the bottom of the table. This format is similar to that of the CUTG database (*See also* **Note 1**).

Monte Carlo Histogram Plots: It is often desirable to compare the properties of some subset of the genes in a genome to the properties of the genome as a whole. For example, one may wish to determine whether the CAI values for genes in a proposed genomic island are significantly lower than those for the genome as a whole. However, visually comparing histograms of CAIs for the island and for the genome can lead to a false inference of difference simply because a small set of genes will tend to have more variation in compositional properties than the genome as a whole. Thus it is useful to be able to test whether observed differences are significant or if they could be explained simply by the reduced size of the examined sub-sample of the genome.

CodonExplorer provides a Monte Carlo method for evaluating the significance of differences between a set of genes and the genome as a whole. The procedure involves generating many (10, 100, or 1,000, at the user's option) random sets of a number of genes equal to those in the selected subset (chosen individually or as a block of contiguous genes). The mean value of the property is then recorded for each randomly chosen set of genes. Finally a

Table 10.1
Compositional properties calculated by CodonExplorer. Arbitrary combinations of these values may be plotted against one another using the "Custom scatter plot" option

Name	Equation/reference
CAI* – the codon adaptation index	$CAI = CAI_{obs}/CAI_{max}$ $CAI_{obs} = \left(\prod_{k=1}^{L} RSCU_k\right)^{(1/L)}$ $CAI_{max} = \left(\prod_{k=1}^{L} RSCU_{k\,max}\right)^{(1/L)}$ $RSCU_{ij} = \frac{x_{ij}}{\frac{1}{n_i}\sum_{j=1}^{n_i} x_{ij}}$ *(14, 43)*
P_3 – The third position GC content	$P_3 = \frac{G_3 + C_3}{A_3 + T_3 + C_3 + G_3}$
P_3/CAI	P_3/CAI (*see* equations for P_3 and CAI, above)
Length	Gene length
Hydrophobicity	Predicted peptide hydrophobicity *(44)*
Horizontal transfer index (HTI)	$Pr(COD_i\|F) = \frac{Pr(F\|COD_i)\,Pr(COD_i)}{\sum_{m=1}^{6} Pr(F\|COD_m)\,Pr(COD_m) + Pr(F\|NON)\,Pr(NON)}$ $(m = 1,2,3,4,5,6)$ The probability that a gene was produced by a Markov model matching in-frame coding genes in a genome (rather than other frames of coding genes or non-coding sequence). Genes with low HTIs and low ribosomal HTIs (see below) have been proposed as transferred *(30)*
Ribosomal HTI	The posterior probability that a gene matches the model for in-frame ribosomal proteins in a genome, rather than intergenic regions separating ribosomal genes or the non-coding frames of ribosomal proteins. Calculated as the HTI above, but using ribosomal proteins or intergenic regions separating ribosomal proteins to build the coding and non-coding models *(30)*
Putative alien	A putatively alien gene identified using the scheme of Nakamura et al. *(30)*
Amino acid frequencies	The frequency of individual amino acids, or a group of amino acids within a gene. Available options include A,C,D,E,F,G,H,I,K,L,M,N,P,Q,R,S,T,V,W,Y, (D+E), (K + R + H) , and (L + I + V + M)

*The equation shown for CAI is the traditional equation *(14)*; the other CAI options are variants of this measure that may be useful for specific analyses.

figure is generated showing a histogram of the average values for each random set (**Fig. 10.9**). The blue circle (dark grey here) represents the mean value of the chosen property for these random sets. The red circle represents the average of the chosen property

Combined graphs for STM

Codon Table

4527 gene(s) (1429204 codon(s))

Fields: [triplet] [frequency: per thousand] ([number])

UUU 23.2 (33210)	**UCU** 7.2 (10281)	**UAU** 17.1 (24424)	**UGU** 4.8 (6886)
UUC 15.2 (21684)	**UCC** 10.1 (14442)	**UAC** 11.5 (16412)	**UGC** 6.6 (9481)
UUA 13.2 (18828)	**UCA** 6.2 (8905)	**UAA** 1.9 (2685)	**UGA** 1.0 (1412)
UUG 12.3 (17581)	**UCG** 9.5 (13566)	**UAG** 0.3 (433)	**UGG** 15.2 (21686)
CUU 11.8 (16828)	**CCU** 7.1 (10216)	**CAU** 13.2 (18928)	**CGU** 18.5 (26451)
CUC 10.4 (14901)	**CCC** 6.9 (9910)	**CAC** 9.5 (13624)	**CGC** 23.1 (33072)
CUA 4.9 (7000)	**CCA** 5.7 (8215)	**CAA** 12.7 (18132)	**CGA** 3.6 (5096)
CUG 53.4 (76262)	**CCG** 24.6 (35095)	**CAG** 30.9 (44132)	**CGG** 6.9 (9912)
AUU 29.3 (41863)	**ACU** 6.7 (9556)	**AAU** 17.8 (25426)	**AGU** 7.3 (10467)
AUC 24.3 (34721)	**ACC** 23.3 (33328)	**AAC** 20.1 (28667)	**AGC** 17.4 (24937)
AUA 5.3 (7582)	**ACA** 5.7 (8187)	**AAA** 34.8 (49731)	**AGA** 2.3 (3344)
AUG 27.3 (39072)	**ACG** 18.8 (26910)	**AAG** 11.2 (15996)	**AGG** 1.6 (2335)
GUU 15.4 (22053)	**GCU** 12.7 (18144)	**GAU** 31.6 (45185)	**GGU** 17.3 (24715)
GUC 18.2 (25970)	**GCC** 29.1 (41591)	**GAC** 20.2 (28927)	**GGC** 35.3 (50413)
GUA 11.3 (16134)	**GCA** 12.9 (18500)	**GAA** 35.0 (50085)	**GGA** 8.7 (12443)
GUG 25.2 (35972)	**GCG** 42.4 (60613)	**GAG** 20.6 (29478)	**GGG** 12.0 (17169)

Coding GC 53.98% 1st letter GC 62.31% 2nd letter GC 41.60% 3rd letter GC 58.03%

Fig. 10.8. The codon usage table for all genes in the *Salmonella enterica* serovar Typhimurium LT2 genome, as output by CodonExplorer.

for the query set. Finally, the light red bars represent a histogram of the values of the chosen property in the query set. Comparing the location of the average of the query set (red circle) against the histogram of average values for equally sized random sets (blue bars) allows the user to determine if the set of genes that they selected varies from other random equally sized gene sets more than would be expected from chance. The caption at the top of the Monte Carlo histogram figure lists the number of random gene sets constructed, how many of those random gene sets had a higher value for the chosen property than the query set, and the probability (as assessed using the one-tailed *t*-test) that the query set average could be as different as it was from a normal distribution fit to the distribution of random set averages by chance.

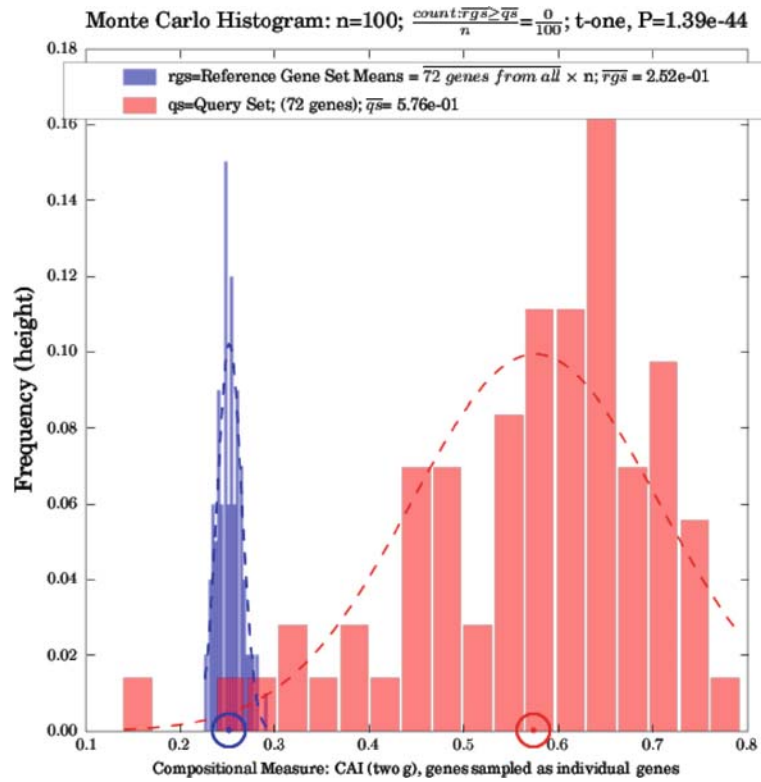


Fig. 10.9. A Monte Carlo histogram in CodonExplorer.

2.5. Starting a New Search

To start over and select a new set of genes for analysis, press the "new search" button at the top right of the analysis screen (your old set of genes will not be saved).

3. Examples

Many classic analyses of codon usage and DNA sequence composition can quickly and easily be recapitulated in CodonExplorer. This section provides a step by step guide to using CodonExplorer to examine the effects of selection, mutation, and horizontal transfer on codon usage.

3.1. Codon Usage as an Indicator of Highly Expressed Genes

Histograms of CAI values, and plots of CAI against GC content at the third codon position (P_3), can provide insight into selection for translational efficiency. Monte Carlo techniques are useful in assessing the significance of differences in CAI or P_3/CAI

between a subset of genes and the genome as a whole. To generate the plots of CAI versus P_3 shown in **Figs. 10.10** and **10.11** requires analysis of all genes in the *Salmonella enterica* serovar Typhimurium LT2 genome (to generate panel A and C in **Fig. 10.10**), followed by an analysis of highly expressed genes in *Salmonella* (to generate panels B and D in **Fig. 10.10** and panels A and B in **Fig. 10.11**).

To select all genes in the *Salmonella enterica* serovar Typhimurium LT2 genome for analysis, enter "stm" (*see* **Note 6**) in the "Enter KEGG genome abbreviation(s)" field.

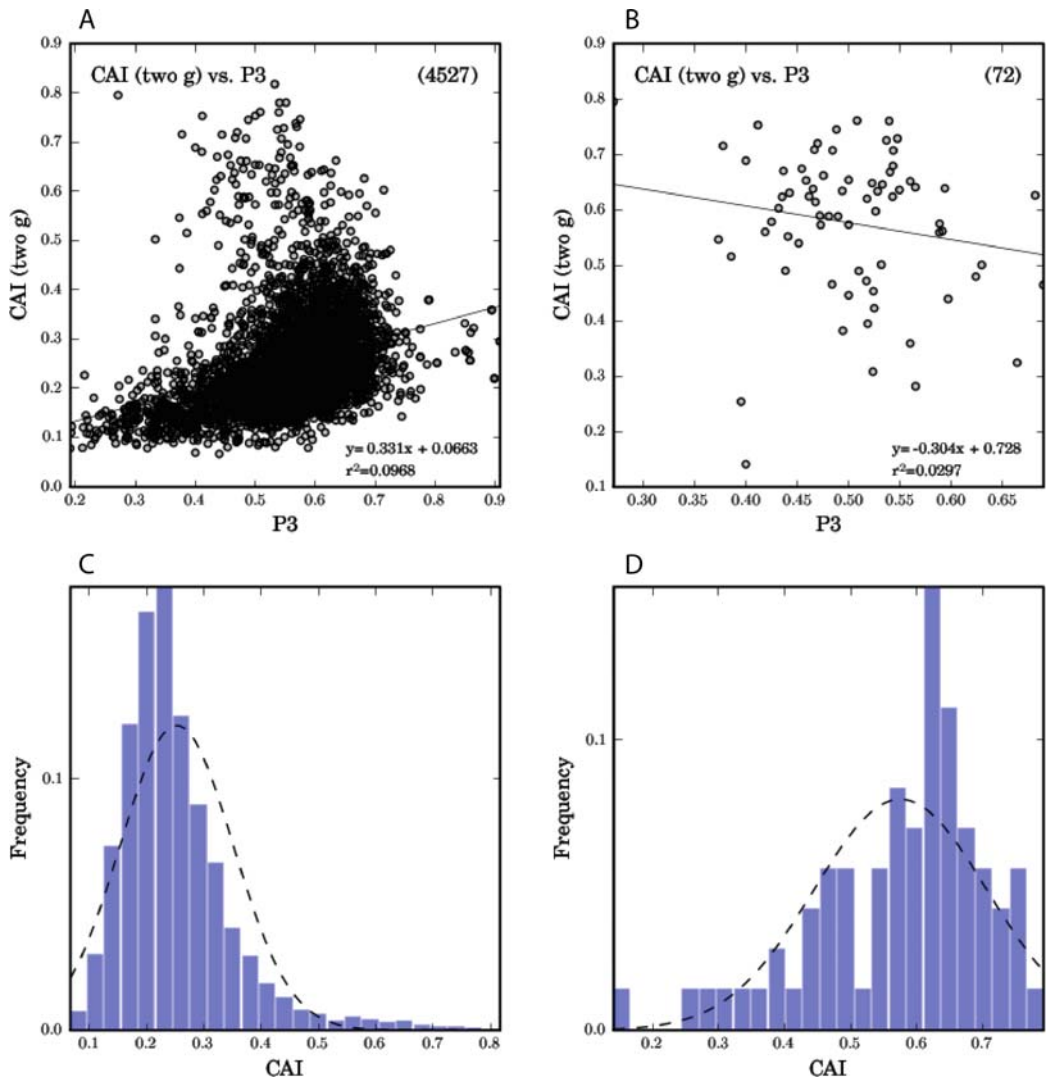


Fig. 10.10. Ribosomal proteins, and proteins from other orthology groups believed to be highly expressed, tend to have higher CAI values than most genes (C + D), and are outliers on a plot of CAI versus P_3 (A + B). In this case, the *Salmonella enterica* serovar Typhimurium genome is plotted.

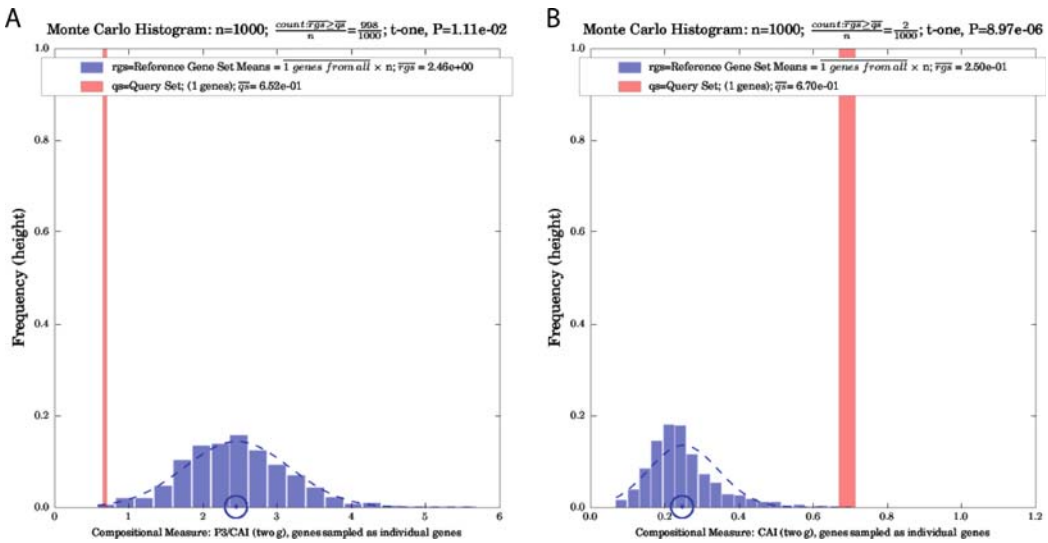


Fig. 10.11. The ribosomal gene coding for the small subunit protein rpsU has significantly high ($p = 0.00000897$) CAI values relative to other genes in *Salmonella enterica* serovar Typhimurium (**A**) , and significantly lower ($p = 0.0111$) P3/CAI values (**B**).

Then press the "Search for Genes" button at the bottom of the page. A new screen will appear showing the group of all genes in *Salmonella,* and several available analyses. Check the "scatter plot" check box by the "CAI vs. P_3" header (to select the analysis in **Fig. 10.10 A**), then check the "CAI" check box to the right of the "Histograms" header (to select the analysis in **Fig. 10.10 C**). Finally, press the "Generate selected graph(s)" button at the bottom of the screen. After a brief wait, the images in **Fig. 10.10 A** and **C** will appear in a new window. (Right-click on the figures and select "Save image as. . ." to save them to disk as .png images.)

When you finish examining and/or saving the figures, close the figure window, and press the "new search button" at the top of the analysis screen to start a new search. On the search screen, locate the "Gene screen" header (**Fig. 10.3**), enter "stm" into the field to the left of the "KEGG genome abbreviation" header, and select "Ribosomal proteins + highly expressed Kos" from the pull-down menu. Press the "Search for Genes" button at the bottom of the screen.

When the analysis screen appears, check the boxes to select a "CAI vs. P3" scatterplot and a CAI histogram as before (to generate **Fig. 10.10 B** and **D**). Finally, to generate **Fig. 10.11 A**, check the "generate Monte Carlo Histograms for the selected gene group(s)" checkbox, then select "CAI (two g)" from the "Compositional measure" pull-down menu, and the desired number of replicates (in this case 100) from the "Number of replicates(n)"

pull-down menu. Set the "Sample reference Set" pull-down menu to "individual genes" (*see* **Note 15**). Then click the "Generate Selected Graph(s)" button. After a brief pause, the images in **Figs.** 10.10 B, D, and **Fig. 10.11 A** should appear in a new window. After examining and (if desired) saving the images to disk, return to the select analysis screen and uncheck all boxes other than the Monte Carlo histogram box. Then set the "Compositional measure" pull-down menu to "P_3/CAI." Once again press "Generate selected graph(s)" at the bottom of the screen. The image from **Fig. 10.11 B** should appear in a new window.

Along with mutation pressure, selection for translational efficiency is believed to be an important force driving codon usage. Thus, those genes that are more highly expressed are expected to have unusually high CAI values for their GC content. Ribosomal proteins are highly expressed (although not all have high CAIs), as are elongation factors and some membrane factors *(27)*. Highly expressed genes can often be detected by plotting CAI versus P_3 (**Fig. 10.10**).

The trends in CAI versus P_3 plots or histograms of CAI can be confirmed using the Monte Carlo histogram feature (**Fig. 10.11**). This is useful because small sets of genes frequently display apparently unusual codon usage or sequence compositional properties simply due to the effects of a small sample size. Thus, comparing against randomly chosen gene sets of the same size can be used to demonstrate or refute the statistical significance of an apparent difference in the plots.

3.2. Codon Usage as an Indicator of Mutation Pressure

CodonExplorer can be used to generate various P_3 versus CAI plots useful for studying the effect of mutation pressure on codon usage. To generate the P_3 versus CAI plots shown in **Fig. 10.12**, enter the KEGG abbreviations for the genomes desired (*see* **Note 6**) in the "Enter KEGG genome abbreviation(s)" field, entering one genome abbreviation per line. (When generating **Fig. 10.12**, "mge," "eco," and "sco" were entered on separate lines to search for all genes in *Mycoplasma genitalium, Escherichia coli* K-12 MG1655, and *Streptomyces coelicolor,* respectively.) Then press the "Search for Genes" button. A new screen will appear listing the genes found by the search, and available analyses. To generate the plots in **Fig. 10.12** click the "scatter plot," "heatmap," and "contour plot" checkboxes next to the "CAI vs. P3" header. Make sure the "Combine Data Sets" option is set to "No," then click the "Generate Selected Graph(s)" button. After a brief wait, figures corresponding to each of the nine panels in **Fig. 10.12** will be generated.

Another type of plot useful for analyzing the effects of mutation pressure is the fingerprint plot *(19)*. To generate the fingerprint plot shown in **Fig. 10.13,** first ensure that you are on the

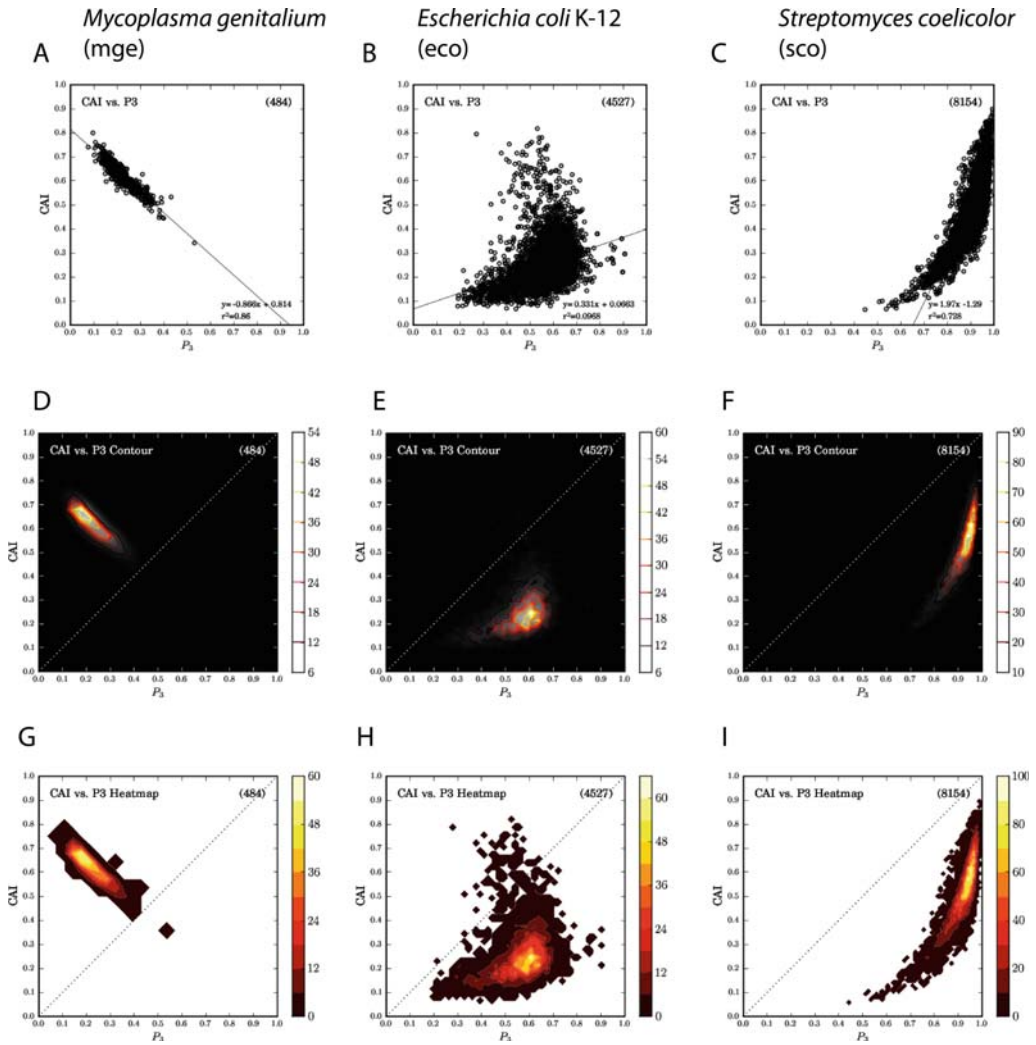


Fig. 10.12. Several views of CAI versus P3 available in CodonExplorer. The genomes chosen reflect low (mge), mid (eco), and high (sco) values of genomic GC content. GC content strongly influences CAI; the two are not independent. Thus, analysis of expression and HGT are connected (Guy and Sueoka, submitted). Heat maps and contour plots show trends that are hard to see in the scatterplot. Seeing which areas are denser allows easier interpretation (e.g., than graphing in Excel and being unable to relate the distribution of points and the regression line).

gene search page (if you are on the select analysis page press the "new search" button at the top of the page to start a new search). Search for all genes in the human genome by entering the KEGG genome id for *Homo sapiens* (hsa) in the "Enter KEGG genome abbreviation(s)" field (*see* **Note 6**). Then press "Search for Genes." A new screen will appear displaying the selected genes and various options for analysis. Select the "combined" and "gradient" checkboxes next to the "Fingerprint plots" header

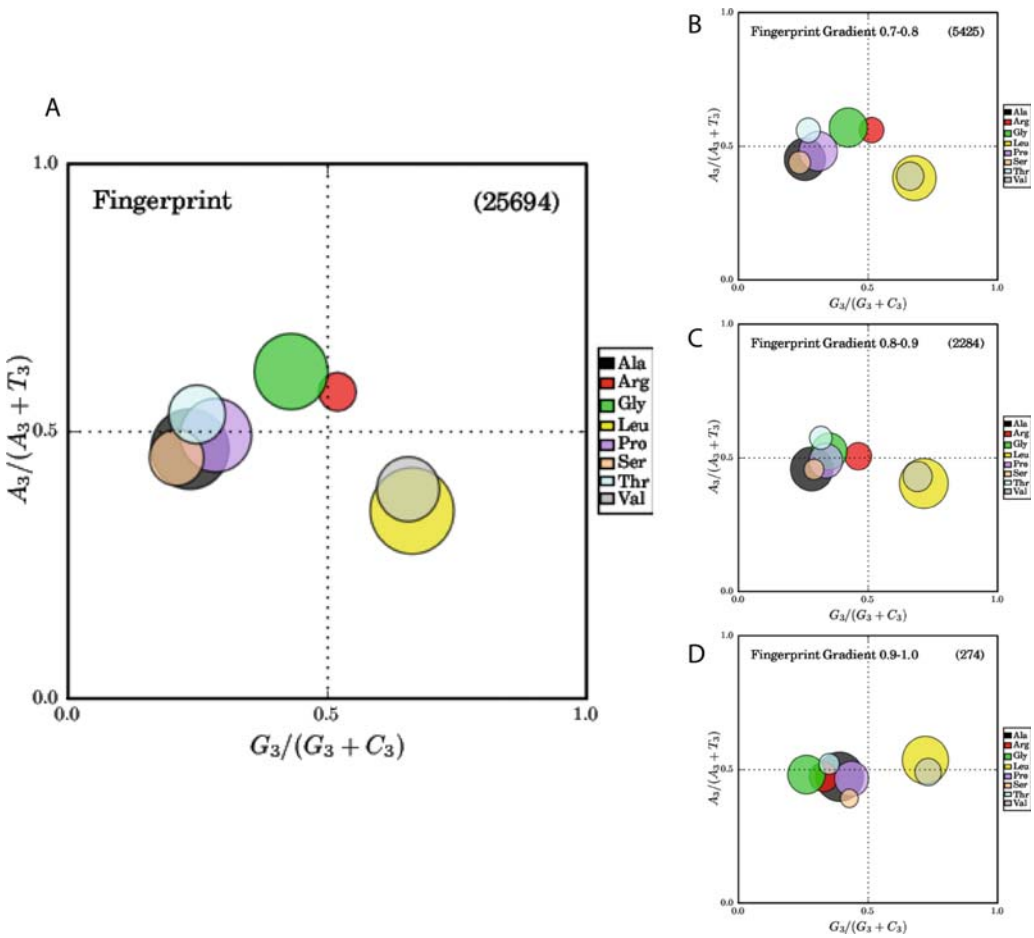


Fig. 10.13. Fingerprint plot of all human genes, similar to that in *(20)*.

in the codon usage section (*see* **Fig. 10.7**). Finally, click the "Generate Selected Graph(s)" button at the bottom of the page. A combined fingerprint plot as well as fingerprint plots for genes in each 10% bin of third codon position GC content (P_3) will be generated. The combined fingerprint plot for *Homo sapiens* (**Fig. 10.13 A**) as well as the fingerprint plots for the 70–80% (**Fig. 10.13 B**), 80–90% (**Fig. 10.13 C**), and 90–100% GC content (**Fig. 10.13 D**) bins for this genome are shown in **Fig. 10.13**.

The scatter plot of CAI versus P_3 illustrates the variable effect of GC content on CAI in different lineages (**Fig. 10.12 A–C**). In the *Mycoplasma genitalium* genome, which possesses a low GC content, there exists a strong negative correlation (r^2 = 0.86) between the position three GC content and CAI, whereas in the high-GC *Streptomyces coelicolor* CAI and GC content are positively correlated (r^2 = 0.73).

One difficulty when examining scatter plots of CAI versus P_3 is that it can be difficult to determine the relative density of points in different regions of a scatter plot when the number of data points is very large. To help solve this problem, CodonExplorer allows generation of both contour plots (**Fig. 10.12 D–F**), and heat maps (**Fig. 10.12 G–I**). These plots more clearly illustrate the relative density of different regions.

The fingerprint plot in **Fig.** 10.13 illustrates that groups of codons containing different nucleotides at the second codon position tend to change in amino/keto bias together *(20)*. The partial gradient fingerprint plot (**Fig.** 10.13 B–D) illustrates that although codons for different amino acids respond differently to changes in GC content, groups of codons with the same second position nucleotide tend to respond similarly.

3.3. Codon Usage and Nucleotide Sequence Composition as Indicators of Horizontal Gene Transfer

Monte Carlo techniques are useful for testing the statistical significance of differences in codon usage or nucleotide sequence composition between putatively transferred sets of genes and the genome as a whole. To generate the Monte Carlo histograms in **Fig. 10.14,** first, select the genomic island of interest from the "select genomic island" pull-down menu (**Fig. 10.3**), or enter the first and last gene in the island of interest, using the "Search by Gene Region" option (**Fig. 10.2**; see Program Usage, above). Then press the "Search for Genes" button at the bottom of the page. Next, on the search results/analysis page, select the "generate Monte Carlo Histograms for the selected gene group(s)"

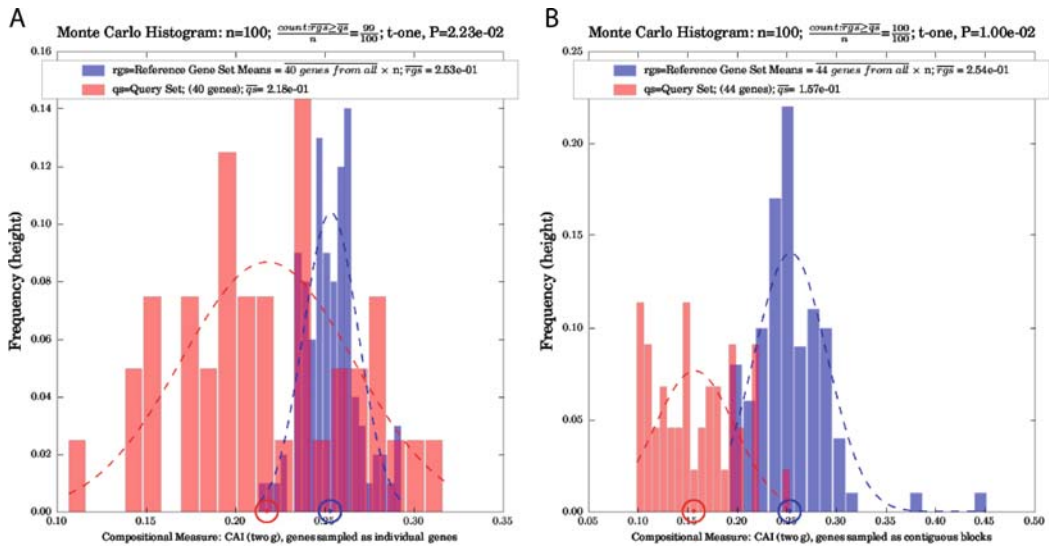


Fig. 10.14. **Monte Carlo histogram of CAI values for the *Salmonella enterica* serovar Typhimurium prophage Fels-1, and the Salmonella Pathogenicity island 2 (SPI-2).** Genes in both the STM Fels-1 prophage (**A**) and SPI-2 (**B**) show significantly (p = 0.0223 and 0.01, respectively) lower average CAI values than the genome as a whole.

checkbox, the desired compositional property from the "compositional measure" pull-down menu (for **Fig. 10.14**, "CAI (two g)" was selected), and the desired number of replicates (in this case 100) from the "Number of replicates(n)" pull-down menu. Make sure the "Contiguous blocks" option is selected from the "Sample reference Set" pull-down menu (*see* **Note 15**) and then click the "Generate Selected Graph(s)" button.

Because optimal codon usage varies in divergent microbial lineages, it has been proposed that genes that have been transferred from one genome to another will have patterns of codon usage more similar to the genome from which they were transferred than to the genome in which they currently reside *(27)* (Guy and Sueoka, submitted). Thus, unusually low CAI values (e.g., as shown for the SPI-2 pathogenicity island and the Fels-1 prophage in the genome of *Salmonella enterica* serovar Typhimurium LT 2 displayed in the Monte Carlo histograms in **Fig. 10.14**) are consistent with the hypothesis that these regions were acquired by HGT. However, these values on their own do not prove that such a transfer occurred.

Although an unusual sequence composition for a gene (or group of genes) is often taken as an indication of horizontal transfer, other factors may also explain unusual codon usage. Thus, while unusual codon usage may suggest the possibility of lateral transfer, the hypothesis of transfer should be confirmed using additional methods for HGT detection, such as discordance between gene trees and organismal trees, lack of orthologs in related lineages, unusual dinucleotide usage, or the presence of nearby mobility genes.

Conversely, horizontally transferred genes may, over time, alter their composition and codon usage to match that of the genome in which they reside *(28)*. Thus a lack of unusual codon usage or nucleotide composition does not rule out the possibility of recent horizontal transfer from a lineage with similar properties, nor does it exclude the possibility of an ancient transfer that is no longer detectable using these methods.

4. Notes

1. Other tools for codon usage analysis are available online. Examples include the following:
   - *CodonW*: A web interface useful for performing correspondence analysis on codon usage values, which can also calculate CAI values and other measures of codon usage bias *(38)*. (http://bioweb.pasteur.fr/seqanal/interfaces/codonw.html).

- *Ade4 and seqinr packages*: seqinr provides a platform for sequence retrieval and analysis using the R statistical package. Ade4 is a package for multivariate statistical analysis and graphical display and allows many graphs related to codon usage to be plotted, including correspondence analysis. The ade4 and seqinr R packages can be run through the Rweb interface (http://pbil.univ-lyon1.fr/Rweb/Rweb.general.html). Documentation for ade4 (http://pbil.univ-lyon1.fr/ADE-4/home.php?lang=eng) *(39)* and seqinr (http://pbil.univ-lyonl.fr/software/seqinr/home.php?lang=eng) is also available.
- *Genometrics Website*: DNA walks for bacteria, archaea, and several plasmid and bacteriophage genomes are available at the Genometrics website (http://www2.unil.ch/comparativegenometrics/DNA_walk.html) *(40)*.
- *Synthetic gene designer*: This tool, available at http://www.evolvingcode.net/codon/sgd/index.php, is designed to help optimize codon usage for heterologus gene expression *(41)*.
- *Codon Usage Tabulated from Genbank (CUTG) database*: (http://www.kazusa.or.jp/codon/) Provides codon usage statistics for sequences for 32,775 organisms as well as for sequences derived from bacteriophage, mitochondria, and chloroplasts. Organisms can be browsed or searched by name *(42)*.

2. One disadvantage to compositional methods for HGT detection is that transfer between related or unrelated strains with similar compositional characteristics will not be detected. Conversely, if a sequence characteristic varies within a genome, then classes of genes, such as ribosomal proteins, that have unusual compositions may appear to have been horizontally transferred. Even when the composition of a transferred gene is initially distinct from the composition of the recipient genome, the composition of the gene will drift toward that of the recipient genome over time. Thus, the traces of ancient transfers may be obliterated in sequences that have had time to equilibrate fully *(28)*.
3. The Kyoto Encyclopedia of Genes and Genomes (KEGG) is an online database that seeks to organize and integrate information on biological systems from the genomic to the environmental level. As part of this effort, KEGG contains re-annotations of genome sequences taken from RefSeq. Each gene within the genomes re-annotated by KEGG is given a unique KEGG gene id. Similarly, each genome is assigned a KEGG genome id. The KEGG website is http://www.genome.jp/kegg/.

4. For details on EC groups, see the website of the Nomenclature Committee of the International Union of Biochemistry and Molecular Biology (http://www.chem.qmul.ac.uk/iubmb/enzyme/).
5. Only genes that match all properties in the fields under search by name or ID will be reported. Thus entering "rpsU" in the gene name field and "stm" (the KEGG genome abbreviation for *Salmonella enterica* serovar Typhimurium LT2) will find only the rpsU gene from *Salmonella*, whereas the same search without a name in the genome field will return records for all genes named "rpsU" across KEGG genomes.
6. A list of KEGG genome IDs, grouped by phylogeny, can be found at the KEGG website (http://www.genome.jp/kegg/catalog/org_list.html). Similarly, a list of NCBI taxon names can be found at http://www.ncbi.nlm.nih.gov/Taxonomy/Browser/wwwtax.cgi?mode=Root.
7. Due to computational constraints, users are currently allowed to select a maximum of 100 genomes at a time for analysis, so selecting large or heavily sampled lineages (e.g., all Gammaproteobacteria) for simultaneous analysis using the NCBI taxonomy search is not possible at this time.
8. Because KO groups may contain multiple members in a single genome (e.g., there are currently two members of KO3553 in the *Bacillus cereus* e331 genome), members of a KO group are not always strict orthologs.
9. If the KO number for a particular group of genes is unknown, but an example of the group is known, one can search for the known gene by name or ID (see above); click "View" (in the view genes column) for a genome containing the gene name or ID, then follow the link under the "description" column to the KEGG gene description page. This description includes the KO group to which the gene belongs.
10. Currently only individual EC IDs are allowed, and not IDs for entire EC groups. However, if all EC IDs within are group are desired, each EC category in the class can be entered on a separate line.
11. In some analyses it is desirable to separate genes on each strand of the DNA. One can do so in CodonExplorer by using the region selector for the entire genome, choosing only genes in either "forward" or "reverse" orientation, then repeating the analysis using genes in the other orientation.
12. The boundaries of many genomic islands are not unambiguously defined and the gene ranges given in the pull-down menu should not be considered definitive. In cases where a better gene range for a gene has been determined (e.g., by

comparative or compositional analysis), literature islands can be expanded or reduced based upon any extra information possessed by the user by searching for the predefined island, noting the first and last genes, and then using the Search by Gene Region feature to define a new region.

13. If searches for genes or genomes repeatedly fail, it is sometimes useful to click the "Clear Search Results" button at the bottom of the page to ensure that all previous entries have been cleared.
14. When plotting codon fingerprints, it is important to remember that the smaller the dataset, the larger the expected visual difference in the plots due to chance. While this is not likely to be a major problem at the whole genome level, it can be an important factor when analyzing codon fingerprints for each gene individually, or when using the P_3 gradient option. It is a good idea to test the significance of an observed visual difference in codon fingerprint plots using Monte Carlo analysis (see the section on Monte Carlo Histograms, above).
15. Comparing a contiguous block of putative HGT genes to individual random genes using the Monte Carlo tool may bias the analysis, since genes are not distributed randomly on the genome. Some classes of genes, such as those encoding ribosomal proteins, are both unusual in composition and clustered together in the genome. Thus it is usually better to compare a contiguous set of genes (e.g., a putative genomic island) to other similarly sized contiguous blocks of genes.

Acknowledgments

This work was supported in part by the Biophysics and SCR training grants T32GM08759 and T32GM065103 from NIH. CodonExplorer is hosted on the W.M. Keck Foundation Bioinformatics Facility at the University of Colorado, Boulder.

References

1. Nirenberg, M. W., and Matthaei, J. H. (1961) The dependence of cell-free protein synthesis in *E. coli* upon naturally occurring or synthetic polyribonucleotides. *Proc Natl Acad Sci USA* **47,** 1588–602.
2. Soll, D., Ohtsuka, E., Jones, D. S., Lohrmann, R., Hayatsu, H., Nishimura, S., and Khorana, H. G. (1965) Studies on polynucleotides, XLIX. Stimulation of the binding of aminoacyl-sRNA's to ribosomes by ribotrinucleotides and a survey of codon assignments for 20 amino acids. *Proc Natl Acad Sci USA* **54,** 1378–85.
3. Sueoka, N. (1961) Compositional correlation between deoxyribonucleic acid and protein. *Cold Spring Harb Symp Quant Biol* **26,** 35–43.

4. Efstratiadis, A., Kafatos, F. C., and Maniatis, T. (1977) The primary structure of rabbit beta-globin mRNA as determined from cloned DNA. *Cell* **10,** 571–85.
5. Sanger, F., Nicklen, S., and Coulson, A. R. (1977) DNA sequencing with chain-terminating inhibitors. *Proc Natl Acad Sci USA* **74,** 5463–7.
6. Sonneborn, T. M. (1965) Nucleotide sequence of a gene: first complete specification. *Science* **148,** 1410.
7. Ikemura, T., and Ozeki, H. (1983) Codon usage and transfer RNA contents: organism-specific codon-choice patterns in reference to the isoacceptor contents. *Cold Spring Harb Symp Quant Biol* **47 Pt 2,** 1087–97.
8. Crick, F. H. (1966) Codon – anticodon pairing: the wobble hypothesis. *J Mol Biol* **19,** 548–55.
9. Agris, P. F., Vendeix, F. A., and Graham, W. D. (2007) tRNA's wobble decoding of the genome: 40 years of modification. *J Mol Biol* **366,** 1–13.
10. Ikemura, T. (1981) Correlation between the abundance of *Escherichia coli* transfer RNAs and the occurrence of the respective codons in its protein genes: a proposal for a synonymous codon choice that is optimal for the *E. coli* translational system. *J Mol Biol* **151,** 389–409.
11. Ikemura, T. (1981) Correlation between the abundance of *Escherichia coli* transfer RNAs and the occurrence of the respective codons in its protein genes. *J Mol Biol* **146,** 1–21.
12. Ikemura, T. (1982) Correlation between the abundance of yeast transfer RNAs and the occurrence of the respective codons in protein genes. Differences in synonymous codon choice patterns of yeast and *Escherichia coli* with reference to the abundance of isoaccepting transfer RNAs. *J Mol Biol* **158,** 573–97.
13. Moriyama, E. N., and Powell, J. R. (1997) Codon usage bias and tRNA abundance in Drosophila. *J Mol Evol* **45,** 514–23.
14. Sharp, P. M., and Li, W. H. (1987) The codon Adaptation Index – a measure of directional synonymous codon usage bias, and its potential applications. *Nucleic Acids Res* **15,** 1281–95.
15. Kane, J. F. (1995) Effects of rare codon clusters on high-level expression of heterologous proteins in *Escherichia coli*. *Curr Opin Biotechnol* **6,** 494–500.
16. Muto, A., and Osawa, S. (1987) The guanine and cytosine content of genomic DNA and bacterial evolution. *Proc Natl Acad Sci USA* **84,** 166–9.
17. Knight, R. D., Freeland, S. J., and Landweber, L. F. (2001) A simple model based on mutation and selection explains trends in codon and amino-acid usage and GC composition within and across genomes. *Genome Biol* **2,** RESEARCH0010.
18. Gupta, S. K., and Ghosh, T. C. (2001) Gene expressivity is the main factor in dictating the codon usage variation among the genes in Pseudomonas aeruginosa. *Gene* **273,** 63–70.
19. Sueoka, N. (1999) Translation-coupled violation of Parity Rule 2 in human genes is not the cause of heterogeneity of the DNA G+C content of third codon position. *Gene* **238,** 53–8.
20. Sueoka, N. (2002) Wide intra-genomic G+C heterogeneity in human and chicken is mainly due to strand-symmetric directional mutation pressures: dGTP-oxidation and symmetric cytosine-deamination hypotheses. *Gene* **300,** 141–54.
21. Sueoka, N. (1988) Directional mutation pressure and neutral molecular evolution. *Proc Natl Acad Sci USA* **85,** 2653–7.
22. Bernardi, G. (1993) The vertebrate genome: isochores and evolution. *Mol Biol Evol* **10,** 186–204.
23. Costantini, M., Clay, O., Auletta, F., and Bernardi, G. (2006) An isochore map of human chromosomes. *Genome Res* **16,** 536–41.
24. Sueoka, N. (1999) Two aspects of DNA base composition: G+C content and translation-coupled deviation from intra-strand rule of A = T and G = C. *J Mol Evol* **49,** 49–62.
25. Lobry, J. R., and Sueoka, N. (2002) Asymmetric directional mutation pressures in bacteria. *Genome Biol* **3,** RESEARCH0058.
26. Sueoka, N. (1995) Intrastrand parity rules of DNA base composition and usage biases of synonymous codons. *J Mol Evol* **40,** 318–25.
27. Karlin, S., Mrazek, J., and Campbell, A. M. (1998) Codon usages in different gene classes of the *Escherichia coli* genome. *Mol Microbiol* **29,** 1341–55.
28. Lawrence, J. G., and Ochman, H. (1997) Amelioration of bacterial genomes: rates of change and exchange. *J Mol Evol* **44,** 383–97.
29. Groisman, E. A., Sturmoski, M. A., Solomon, F. R., Lin, R., and Ochman, H. (1993) Molecular, functional, and

evolutionary analysis of sequences specific to Salmonella. *Proc Natl Acad Sci USA* **90,** 1033–7.

30. Nakamura, Y., Itoh, T., Matsuda, H., and Gojobori, T. (2004) Biased biological functions of horizontally transferred genes in prokaryotic genomes. *Nat Genet* **36,** 760–6.
31. Lobry, J. R. (1997) Influence of genomic G+C content on average amino-acid composition of proteins from 59 bacterial species. *Gene* **205,** 309–16.
32. Faith, J. J., and Pollock, D. D. (2003) Likelihood analysis of asymmetrical mutation bias gradients in vertebrate mitochondrial genomes. *Genetics* **165,** 735–45.
33. Hacker, J., and Kaper, J. B. (2000) Pathogenicity islands and the evolution of microbes. *Annu Rev Microbiol* **54,** 641–79.
34. Hsiao, W. W., Ung, K., Aeschliman, D., Bryan, J., Finlay, B. B., and Brinkman, F. S. (2005) Evidence of a large novel gene pool associated with prokaryotic genomic islands. *PLoS Genet* **1,** e62.
35. Rudner, R., Karkas, J. D., and Chargaff, E. (1969) Separation of microbial deoxyribonucleic acids into complementary strands. *Proc Natl Acad Sci USA* **63,** 152–9.
36. Sueoka, N. (1962) On the genetic basis of variation and heterogeneity of DNA base composition. *Proc Natl Acad Sci USA* **48,** 582–92.
37. Sharp, P. M., and Devine, K. M. (1989) Codon usage and gene expression level in Dictyostelium discoideum: highly expressed genes do 'prefer' optimal codons. *Nucleic Acids Res* **17,** 5029–39.
38. Peden, J. F. (1999) Analysis of codon usage. (Ph.D. Thesis), Department of Genetics, University of Nottingham, Nottingham, UK.
39. Thioulouse, J., Chessel, D., Dolédec, S., and Olivier, J. M. (1996) ADE-4: a multivariate analysis and graphical display software. . *Stat Comput* **7,** 75–83.
40. Roten, C. A., Gamba, P., Barblan, J. L., and Karamata, D. (2002) Comparative Genometrics (CG): a database dedicated to biometric comparisons of whole genomes. *Nucleic Acids Res* **30,** 142–4.
41. Wu, G., Bashir-Bello, N., and Freeland, S. J. (2006) The synthetic gene designer: a flexible web platform to explore sequence manipulation for heterologous expression. *Protein Expr Purif* **47,** 441–5.
42. Nakamura, Y., Gojobori, T., and Ikemura, T. (1997) Codon usage tabulated from the international DNA sequence databases. *Nucleic Acids Res* **25,** 244–5.
43. Sharp, P. M., and Li, W. H. (1986) Codon usage in regulatory genes in *Escherichia coli* does not reflect selection for 'rare' codons. *Nucleic Acids Res* **14,** 7737–49.
44. Kyte, J., and Doolittle, R. F. (1982) A simple method for displaying the hydropathic character of a protein. *J Mol Biol* **157,** 105–32.

Chapter 11

Genetic Code Prediction for Metazoan Mitochondria with GenDecoder

Federico Abascal, Rafael Zardoya, and David Posada

Abstract

There is a standard genetic code that is used by most organisms, but exceptions exist in which particular codons are translated with a different meaning, i.e., as a different amino acid. The characterization of the genetic code of an organism is hence a key step for properly analyzing and translating its protein-coding genes. Such characterization is particularly important in the case of metazoan mitochondrial genomes for two reasons: first, many variant codes occur in them and second, mitochondrial data is frequently used for evolutionary studies. Variant codes are usually found by comparative sequence analyses. Given a protein alignment, if a particular codon for a given species occurs at positions in which a particular amino acid is frequently found in other species, then the most likely hypothesis is that the codon is translated as that particular amino acid in that species. Previously, we have shown that this method can be very reliable provided that enough taxa and positions are included in the comparisons and have implemented it in the web-ser GenDecoder (http://darwin.uvigo.es/software/gendecoder.html).

In this chapter we describe the rationale of the method used by GenDecoder and its usage through worked examples, highlighting the potential problems that can arise during the analysis.

Key words: Genetic code, metazoan, mitochondria, genomics, bioinformatics.

1. Introduction

The genetic code of an organism provides the translation table between the languages in which DNA and proteins are coded. In this table a correspondence between each specific nucleotide triplet (codon) and each amino acid is established. A relevant property of the genetic code is that it is nearly universal, i.e., distantly related organisms such as *E. coli* and humans share the same code, indicating that the genetic code was established soon in the history

David Posada (ed.), *Bioinformatics for DNA Sequence Analysis,* Methods in Molecular Biology 537

DOI 10.1007/978-1-59745-251-9_11

of life on Earth, before the split of the three main kingdoms *(1)*. The form of the standard code is not random, since it has been demonstrated that minimizes the impact of mutations and also that it is correlated with the pathways of amino acid biosynthesis (reviewed in ref. *2*). Interestingly, although the standard genetic code is highly optimal, multiple variants have been identified in several nuclear and organellar systems (reviewed in ref. *3*), and are particularly frequent in metazoan mitochondria (*mt*) where up to 11 variants have been already described *(4)*. In these variants one or more codons have been reassigned to a different amino acid, and these changes are usually explained by mutations in tRNAs and/or rRNAs *(5, 6)*. The high frequency of reassignments in metazoan mt-genomes is likely due to the pressure toward compacting the genome, and consequently toward a smaller repertoire of tRNA genes *(7)*. In parallel, as the size of the mt-genomes becomes smaller, the number of times a codon is used decreases (or the codon even disappears), and hence reassignments have a lower mutational impact on the system.

Although initial genetic codes were elucidated through elegant and seminal in vitro experiments, the wealth of sequence data available today allows for the reliable determination of the existing alternative genetic codes using exclusively comparative studies *(8–10)*. The rationale is the following: if a particular codon of a given species occurs at (homologous) protein sites in which a given amino acid is frequently found in related species (i.e., it is conserved), then that particular codon most likely translates as that amino acid. As will be discussed below, although the methodology has demonstrated highly precise *(11)*, some caution must be taken when applying this method of inference under some scenarios, for example, when the problem and reference species are too divergent. Although, the sequence comparisons can be applied by hand, statistical confidence of genetic code inferences increase with the number of taxa and positions compared, and hence use of computers is necessary. Nowadays, the only available program that implements this methodology is GenDecoder (http://darwin.uvigo.es/software/gendecoder.html), which is restricted to the analysis of complete metazoan mt-genomes *(11)*. In this chapter we will describe the usage of this tool with the help of some guided examples.

2. Materials

The only input that GenDecoder requires is a metazoan mt-genome, either supplied as an NCBI's taxonomic identifier or as a GenBank formatted file *(12)*:

(a) If the mt-genome of the species to be analyzed is already deposited in GenBank, the easiest procedure is to supply the corresponding species taxonomic identifier, which can be found either in GenBank or in the NCBI Taxonomy database (http://www.ncbi.nlm.nih.gov/). For example, "7277" is the NCBI TaxID for *Drosophila melanogaster*.

(b) Users who have sequenced a mt-genome and want to characterize its genetic code before submitting it to public databases (what should be a common practice in order to avoid erroneous genetic code annotations), can send GenBank formatted files containing the mt-genome to the GenDecoder web-server. Such files can be prepared with the software Sequin, which is available at http://www.ncbi.nlm.nih.gov/Sequin/index.html. It is a requirement that all the protein-coding genes are properly annotated, and that the annotations follow gene-name conventions (*see* **Note 1**). In addition, a translation table (i.e., a genetic code) must be indicated (*see*-**Note 2**).

The starting page of the GenDecoder web-server is shown in **Fig. 11.1**.

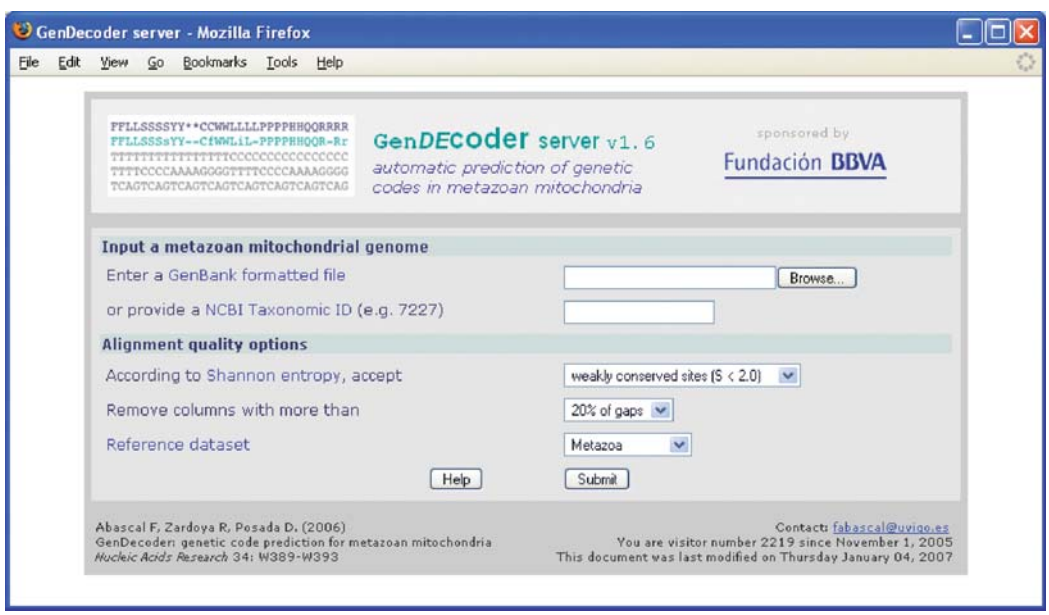


Fig. 11.1. **The GenDecoder web-server starting page.** A metazoan mt-genome must be provided in the first section. Optionally, the user can modify the parameters related to the quality of the alignments (entropy and percentage of gaps accepted) and the reference datasets (Metazoa, Platyhelminths, and Nematoda) in the second section.

3. Methods

Understanding the methodology underlying GenDecoder will help the user to interpret its results. The following work flow is based on **Fig. 11.2**.

1. For a given metazoan species, its mt-genome is retrieved and each of its (usually) 13 protein-coding genes obtained.
2. Each gene is then translated to amino acids according to the expected genetic code (not necessarily the true one) and aligned to its orthologs in other metazoan species. This results in 13 multiple alignments of proteins.
3. The nucleotide sequence of each gene is scanned in order to identify the positions at which each codon occurs. These occurrences are then mapped onto the multiple alignments of proteins. In **Fig. 11.2** the particular case of the UCU codon is illustrated.

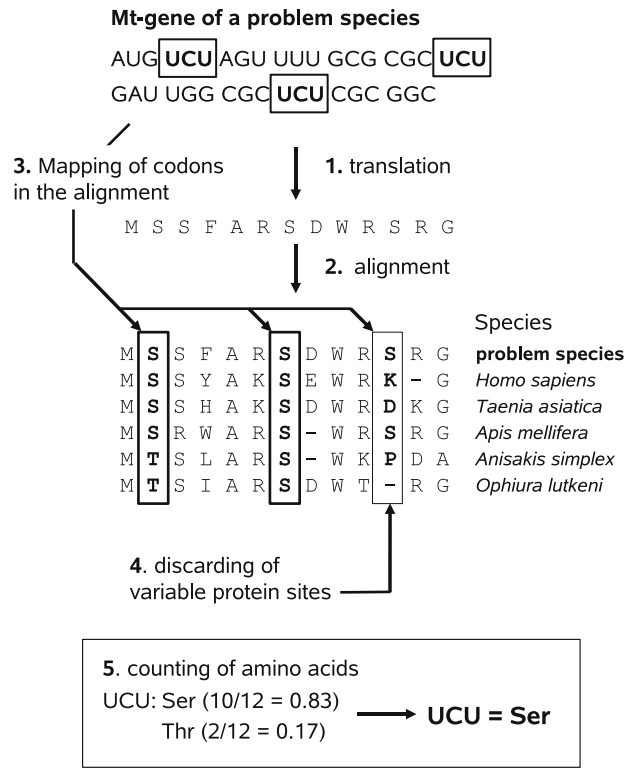


Fig. 11.2. **Work flow of GenDecoder's method for genetic code prediction.** The particular case of the UCU codon is illustrated as an example. A similar pipeline is applied for each of the 64 codons at each of the (usually) 13 protein coding mt-genes.

4. Then, alignment positions that either are too variable or contain too many gaps are filtered out, since they may introduce noise in the analysis. Positions kept are called "informative." Different criteria, based on Shannon's entropy *(13)* and the percentage of gaps, are available in GenDecoder to define "noisy" positions.
5. Finally, for each codon the frequency of the different amino acids at the protein sites in which they occur is measured. The predicted translation of the codon is the one corresponding to the most frequent amino acid. For example, in **Fig.** 11.2 the codon UCU appears at two informative protein sites (the third appearance is filtered out in Step 4) and most likely translates as the amino acid Ser.

We have already demonstrated that this methodology is very accurate (*11*), reaching precision values of 99%. In the sections that follow, we work through some examples to illustrate the usage of GenDecoder and the interpretation of its results, highlighting those cases in which errors might be more likely.

3.1. GenDecoder's Output

The meaning of the main results section of GenDecoder should be straightforward. The line labeled as "*Expected*" corresponds to the expected genetic code as annotated in GenBank. The second line (*Predicted*) corresponds to the code of the query species as predicted by GenDecoder. Here, cases in which the prediction is based on less than four codons or cases in which the difference between the frequency of the predicted and expected amino acids is lower than 0.25 are indicated in lowercase to help identifying unreliable predictions. When a codon is not present in the query species, a dash symbol (–) appears. Similarly, if a codon occurs in the mt-genome, but not at informative positions, its meaning is not predicted and such incidence is indicated with a question mark (?). It is possible to inspect which are the alignment positions responsible of each prediction by clicking on the predicted amino acids. In the next three lines the three positions of every codon are depicted. Below, some lines inform the user about the occurrence of codons at informative positions (*Codon-imp*), their abundance (*Codon-num*), the frequency of the predicted amino acid ("*Freq-aa*"; only the first decimal is shown), and the difference between the frequency of the predicted and expected amino acids (*Diff-freq*). This information might help determining whether a given assignment is reliable. Finally, the nucleotide usage and the "*effective number of codons*" *(14)* is written.

In addition to the main result section of GenDecoder, more detailed information, including the number of occurrences of each codon and the number of positions identified as "noisy" and "informative" in each alignment, is written. The alignments of the mt-proteins can also be examined with the JalView alignment editor (*15*).

4. Examples

4.1. Characterization of the Genetic Code of Speleonectes tulumensis (Crustacea, Remipedia)

The mt-genome of the arthropod species of *Speleonectes tulumensis (16)* is already deposited in GenBank and it is annotated at the moment of writing this chapter as having the Invertebrate Mitochondrial Genetic Code (translation table number 5). The NCBI taxonomic identifier is 84346, which we will use as input for GenDecoder (http://darwin.uvigo.es/software/gendecoder.html) leaving all other options with the default settings (alignment positions either with Shannon entropy values higher than 2.0 or with more than a 20% of gaps are filtered out).

After a while, not more than a couple of minutes, the output of GenDecoder is printed out to the browser window. Some information referred to the alignments, the number of positions filtered, and the codon-usage is displayed first. Alignments can be inspected with the help of the program Jalview *(15)* by clicking in the "get alignment" link next to the gene names. The main result of GenDecoder is at the end of the page (**Fig. 11.3a**). Two predictions (highlighted in red) are different from the annotations in GenBank:

(1) The AGC codon is predicted as Gly instead of Ser. The AGC codon occurs 13 times in the mt-genome of *S. tulumensis*, but only three instances occur at alignment positions below the default conservancy threshold. Hence, this prediction is based on three codon occurrences only. In the "Freq-aa" line we can see that the frequency of Gly is only around 0.3 (previously, in the results page, it was indicated that the frequency is exactly 0.32), and in the line "Diff-freq" it is indicated that this frequency is only about 0.1 larger than the frequency of the expected amino acid (Ser, 0.24). By clicking on the red "g" (*see* **Note 3**) we can see the three alignment positions at which AGC appear (**Fig. 11.3b**): twice in the COX1 alignment (positions 56 and 167) and once in the ND4 alignment (position 174). The poor support for the AGC assignment (low number of codons and low frequency of the predicted amino acid) suggests that this is an unreliable prediction. In fact, if we run GenDecoder with a softer threshold to include "variable sites ($S < 3.0$)," then the number of AGC occurrences increases to six and the corresponding prediction becomes Ser, as expected (*see* **Note 4**).

(2) The AGG codon is predicted as Lys instead of Ser. In this case, the prediction is based on nine codon occurrences. The frequency of Lys at those positions is 0.87, indicating that the prediction is based on protein sites that are strongly conserved along the history of metazoans. Moreover, the Ser amino acid (the expected meaning of AGG) is very rare at those positions (indicated in the "Diff-freq" line). Hence, this is a highly supported prediction likely corresponding to a new genetic code (*see* ref. *4* for details).

a

```
------------------------------------------------------------------------
             MAIN RESULT OF GENDECODER [help]
------------------------------------------------------------------------

Species  : Speleonectes tulumensis
Expected : FFLLSSSSYY**CCWWLLLLPPPPHHQQRRRRIIMMTTTTNNKKSSSSVVVVAAAADDEEGGGG
Predicted: FFLLSSSSYY--CCWWLLLlPPPPHHQQRrRRIIMMTTTTNNKKSgSKVVVVAAAADDEEGGGG
First    : TTTTTTTTTTTTTTTTCCCCCCCCCCCCCCCCAAAAAAAAAAAAAAAAGGGGGGGGGGGGGGGG
Second   : TTTTCCCCAAAAGGGGTTTTCCCCAAAAGGGGTTTTCCCCAAAAGGGGTTTTCCCCAAAAGGGG
Third    : TCAGTCAGTCAGTCAGTCAGTCAGTCAGTCAGTCAGTCAGTCAGTCAGTCAGTCAGTCAG

Codon-imp: +++++++8++0085+++7+3+++9+++9+3+6+++++++4+++473+9+++7+++4++++++++
Codon-num: ++++++++++00++++++++++++++++4+6++++++++++++++++++++++++++++++++
Freq-aa  : 7877676579--47977778998989789999553576657798636947668777999978 98
Diff-freq: ooooooooo--ooooooooooooooooooooooooooooooo1o8ooooooooooooooo
Total number of codons: 3674

Nucleotide usage:
     Total  : A: 31%   T: 36%   G: 16%   C: 17%   GC: 33%
     1st pos: A: 33%   T: 30%   G: 21%   C: 16%   GC: 37%
     2nd pos: A: 18%   T: 46%   G: 15%   C: 21%   GC: 36%
     3rd pos: A: 43%   T: 31%   G: 11%   C: 15%   GC: 26%
```

b

```
Gene: COX3
Column  56: ITTTLTTTSTTTTTVTTTITTTITLTTTVTTTTLIMSTNTTVTTSTTILTTTTTI
Column 167: GSSSGAASFASGSALGAWGASSGAGSASSAGDSFSSGSSSASASSSAGFSSSSGS

Gene: ND4
Column 174: GGGGFSGGSGGGGGSCGGGGGGSGGGGGSGGSGGFSGGSGGGGGSGGSGGGGGGG
```

Fig. 11.3. **GenDecoder results for *S. tulumensis*.** In panel (**a**) the main result of GenDecoder is shown. Codon-imp, number of codons at conserved positions (in this case S < 2.0, gaps < 20%); Codon-num, number of codons in the mt-genome; Freq-aa, first decimal in the frequency of the most frequent amino acid; Diff-freq, difference between the frequency of the predicted and expected amino acids. In panel (**b**) the three alignment positions supporting the prediction of the AGC codon are shown.

4.2. Characterization of the Genetic Code of Hymenolepis diminuta *(Platyhelminthes, Cestoda)*

The species *Hymenolepis diminuta* corresponds to the platyhelminth rat tapeworm, and the genetic code annotated for its mitochondrial genome is the Echinoderm and Flatworm Mitochondrial Code (translation table 9) *(17)*. The NCBI taxonomic identifier of this species is 6216, representing the only input required for an initial run of GenDecoder. The corresponding results indicate that up to six codons have predictions different from the expected code (**Fig. 11.4a**). By carefully inspecting these predictions in a similar manner as described in the previous example, we can determine that all these predictions correspond either to assignments supported by very few codon occurrences, like in the case of the CTC and CGC codons (just one codon occurrence for each of them), or to assignments based on amino acid frequencies that are neither high nor much larger than the frequency of the expected amino acid (TTC, TGT, GTG, and GCG codons, which have "Diff-freq" values close to zero). Hence, in this case all non concordant predictions seem unreliable. The codon CGG is not

a

```
------------------------------------------------------------------------
             MAIN RESULT OF GENDECODER [help]
------------------------------------------------------------------------

Species  : Hymenolepis diminuta
Expected : FFLLSSSSYY**CCWWLLLLPPPPHHQQRRRRIIIMTTTTNNNKSSSSVVVVAAAADDEEGGGG
Predicted: FlLLSSSSYY--sCWWLmLLPPPpHHQQRgR?IIIMTtTTNNNKSSSSVVVlAAAlDDEEGGGG
First    : TTTTTTTTTTTTTTTTCCCCCCCCCCCCCCCCAAAAAAAAAAAAAAAAGGGGGGGGGGGGGGGG
Second   : TTTTCCCCAAAAGGGGTTTTCCCCAAAAGGGGTTTTCCCCAAAAGGGGTTTTCCCCAAAAGGGG
Third    : TCAGTCAGTCAGTCAGTCAGTCAGTCAGTCAGTCAGTCAGTCAGTCAGTCAGTCAGTCAG

Codon-imp: +++++5++++00+8+++1+9+5+3+84++170+8+++2+++5+++7+++5+++7+6+6+++6++
Codon-num: +++++7++++00+++++4+++6+3++7++1+1+++++5++++++++++9+++7++++++++++
Freq-aa  : 4355363544--2276693789797957787-34344544433834463522445378687+86
Diff-freq: ooooooooooo--1oooo9ooooooooooo7o-ooooooooooooooooooooooooo1oooooooo
```

b

```
------------------------------------------------------------------------
             MAIN RESULT OF GENDECODER [help]
------------------------------------------------------------------------

Species  : Hymenolepis diminuta
Expected : FFLLSSSSYY**CCWWLLLLPPPPHHQQRRRRIIIMTTTTNNNKSSSSVVVVAAAADDEEGGGG
Predicted: FFLLSsSSYY--aCWWLmlLPPPpHHqQR?R?IIIMTtTTNNNKSSSSVvVVAAA?DDEEGGGG
First    : TTTTTTTTTTTTTTTTCCCCCCCCCCCCCCCCCAAAAAAAAAAAAAAAAGGGGGGGGGGGGGGGG
Second   : TTTTCCCCAAAAGGGGTTTTCCCCAAAAGGGGTTTTCCCCAAAAGGGGTTTTCCCCAAAAGGGG
Third    : TCAGTCAGTCAGTCAGTCAGTCAGTCAGTCAGTCAGTCAGTCAGTCAGTCAGTCAGTCAG

Codon-imp: +7+++367+800+4++9125+4+2+729+050+4+++298+48++7+++2+++5+0+6+++6++
Codon-num: +++++7++++00+++++4+++6+3++7++1+1+++++5++++++++++9+++7++++++++++
Freq-aa  : 6467475666--239789+79+8+8+589-9-45356565535954585923556-88899+98
Diff-freq: ooooooooooo--ooooo9ooooooooooo-o-oooooooooooooooooooooo-oooooooo
```

Fig. 11.4. **GenDecoder results for *H. dimidiata***. The outputs of GenDecoder after a run with default parameters (S < 2.0, gaps < 20%) and a more restrictive threshold for filtering out variable positions (S < 1.0, gaps < 20%) are shown in panels (**a**) and (**b**), respectively.

predicted because it occurs just once in the genome of *H. diminuta* at a position that is filtered out according to the threshold used (S < 2.0 and gaps < 20%). Codons which are left unassigned are represented with a question mark.

Interestingly, if we restrict the threshold to include only "highly conserved sites (S < 1.0)," then the number of unreliable assignments is reduced to 2 (**Fig. 11.4b**), indicating that the source of erroneous annotations is most likely related to the inclusion of variable alignment positions. Interestingly, platyhelminths (and also nematodes) are particularly sensible to this threshold restriction because they are highly divergent in terms of sequence information with respect to the other metazoans. If the problem species is highly divergent to the reference species, then fewer sites will be conserved among those species, leading to less trusted predictions. To help correct this potential problem, GenDecoder has two alternative reference species data sets comprising 10 and 12 species of Platyhelminthes and Nematoda, respectively, which perform better when platyhelminths and nematodes are analyzed.

In fact, if we run GenDecoder supplying the 6216 NCBI taxonomic identifier, selecting the Platyhelminthes reference data set and leaving all the other options as default, then no erroneous assignment is predicted (results not shown). This example of the rat tapeworm clearly shows that comparing the appropriate species is important to obtain trustworthy predictions of the genetic code.

5. Notes

1. To automatically identify gene orthology relationships, GenBank files must be annotated according to gene name conventions. For instance, "cytochrome c oxidase subunit I" should be named as COX1 or CO1, "NADH dehydrogenase subunit 1" as ND1, and "ATP synthase F0 subunit 8" as ATP8.
2. It is not relevant whether the annotated genetic code for the query species is the true one (in fact, this is what we want to ascertain). However some starting point is required in order to be able to translate protein-coding genes and successfully align them to their orthologs in other species. To provide a genetic code that is as close as possible to reality, we recommend that the genetic code of the closest species available in GenBank is used as an initial guess.
3. As a rule of thumb aimed to highlight potentially unreliable predictions, assignments are typed using lowercase when there are less than four codon observations or when the difference in the frequency of the most frequent amino acid is not sufficiently larger (at least 0.25) than the frequency of the expected amino acid (if the predicted and expected amino acids are not the same). The absence of a codon in the mtgenome it is indicated by a dash symbol (–). Alternatively, if a codon is present but not at alignments columns for which the conservancy threshold holds, then its meaning is not predicted and such occurrence is reported using a question mark (?).
4. It is recommended to test the consistency across different conservancy thresholds of those assignments that are nonconcordant to GenBank. A restrictive threshold will imply that each codon occurrence will be mapped to more conserved protein sites, i.e., protein sites that are more informative for the assignment of the meaning of the codon. However, such restrictive thresholds will reduce the number of occurrences of each codon, which is undesirable from a statistical point of view. Hence, some balance is desired,

particularly for codons that are infrequent. We suggest that the results obtained under different thresholds are compared. If the prediction is the same across different settings, then it becomes more reliable.

References

1. Crick, F. H. (1968) The origin of the genetic code *J Mol Biol*, **38,** 367–79.
2. Di Giulio, M. (2005) The origin of the genetic code: theories and their relationships, a review. *Biosystems* **80,** 175–84.
3. Knight, R. D., Freeland, S. J., and Landweber, L. F. (2001) Rewiring the keyboard: evolvability of the genetic code. *Nat Rev Genet* **2,** 49–58.
4. Abascal, F., Posada, D., Knight, R. D., and Zardoya, R. (2006) Parallel evolution of the genetic code in arthropod mitochondrial genomes *PLoS Biol* **4,** e127.
5. Yokobori, S., Suzuki, T., and Watanabe, K. (2001) Genetic code variations in mitochondria: tRNA as a major determinant of genetic code plasticity. *J Mol Evol* **53,** 314–26.
6. Ivanov, V., Beniaminov, A., Mikheyev, A., and Minyat, E. (2001) A mechanism for stop codon recognition by the ribosome: a bioinformatic approach. *RNA* **7,** 1683–92.
7. Knight, R. D., Landweber, L. F., and Yarus, M. (2001) How mitochondria redefine the code. *J Mol Evol* **53,** 299–313.
8. Beagley, C. T., Okimoto, R., and Wolstenholme, D. R. (1998) The mitochondrial genome of the sea anemone Metridium senile (Cnidaria): introns, a paucity of tRNA genes, and a near-standard genetic code. *Genetics* **148,** 1091–108.
9. Barrell, B. G., Bankier, A. T., and Drouin, J. (1979) A different genetic code in human mitochondria. *Nature* **282,** 189–94.
10. Telford, M. J., Herniou, E. A., Russell, R. B., and Littlewood, D. T. (2000) Changes in mitochondrial genetic codes as phylogenetic characters: two examples from the flatworms. *Proc Natl Acad Sci USA* **97,** 11359–64.
11. Abascal, F., Zardoya, R., and Posada, D. (2006) GenDecoder: genetic code prediction for metazoan mitochondria. *Nucleic Acids Res* **34,** W389–93.
12. Benson, D. A., Karsch-Mizrachi, I., Lipman, D. J., Ostell, J., and Wheeler, D. L. (2004) GenBank: update. *Nucleic Acids Res* **32,** D23–6.
13. Shannon, C. E. (1997) The mathematical theory of communication. 1963 *MD Comput* **14,** 306–17.
14. Wright, F. (1990) The 'effective number of codons' used in a gene. *Gene* **87,** 23–9.
15. Clamp, M., Cuff, J., Searle, S. M., and Barton, G. J. (2004) The Jalview Java alignment editor. *Bioinformatics* **20,** 426–7.
16. Lavrov, D. V., Brown, W. M., and Boore, J. L. (2004) Phylogenetic position of the Pentastomida and (pan)crustacean relationships. *Proc Biol Sci* **271,** 537–44.
17. von Nickisch-Rosenegk, M., Brown, W. M., and Boore, J. L. (2001) Complete sequence of the mitochondrial genome of the tapeworm Hymenolepis diminuta: gene arrangements indicate that Platyhelminths are Eutrochozoans *Mol Biol Evol* **18,** 721–30.

Chapter 12

Computational Gene Annotation in New Genome Assemblies Using GeneID

Enrique Blanco and Josep F. Abril

Abstract

The sequence of many eukaryotic genomes is nowadays available from a personal computer to any researcher in the world-wide scientific community. However, the sequences are worthless without the adequate annotation of the biological meaningful elements. The annotation of the genes, in particular, is a challenging task that can not be tackled without the aid of specific bioinformatics tools. We present in this chapter a simple protocol mainly based on the combination of the program GeneID and other computational tools to annotate the location of a gene, which was previously annotated in *D. melanogaster*, in the recently assembled genome of *D. yakuba*.

Key words: Gene finding, gene, protein, genome, exon, intron, sequence alignment, annotation pipeline.

1. Introduction

One of the major problems that biologists have ever faced is how to extract relevant information from millions of nucleotides produced by large-scale genome sequencing projects *(1)*. The first task is usually the elaboration of the catalogue of protein-coding genes encoded in the genomic sequence *(2)*. The identification of genes in eukaryotes is, however, difficult because of many reasons. The genes are embedded into large intergenic regions along the genome, existing areas with different gene density. In addition, the cellular splicing machinery removes from the gene transcripts those segments that do not contain useful information for protein synthesis (introns), just assembling the coding fragments (exons) that will be translated in the ribosomes (*see 3* for a review on gene structure).

David Posada (ed.), *Bioinformatics for DNA Sequence Analysis,* Methods in Molecular Biology 537

DOI 10.1007/978-1-59745-251-9_12

Finally, both ends of the gene exon assembly include non-coding information (untranslated regions, UTR), while the rest of the gene constitutes the protein-coding sequence (CDS). In fact, less than 2% of the human genome is estimated to code for proteins *(4)*.

Computational recognition of genes is based on (a) the detection of short signals imprinted in the genomic sequence that are involved in exon definition (search by signal), (b) the statistical analysis of genomic regions to distinguish coding from non-coding sequences (search by content), and (c) the detection of similar sequences whose correct distribution of exons and introns has been already established (search by homology). In general, most gene-prediction tools can only identify the coding exons of the gene. Experimental techniques to sequence full transcripts (cDNA) or fragments of them (expressed sequence tags, ESTs) are, thus, more appropriate to accurately annotate the 5′UTR and the 3′UTR of the genes *(5)*. Recently, new integrative tools are incorporating mRNA and EST evidences, though, to provide predictions that include such data *(6, 7)*. Once the draft of the sequence of a new genome is available, a computational pipeline mainly consisting on the implementation of the previous steps must process the sequence in order to provide the gene annotations that are accessible through the major genome browsers such as UCSC *(8)*, Ensembl *(9)*, or NCBI *(10)*.

The program GeneID was one of the first programs to predict full exonic structures of vertebrate genes in anonymous DNA sequences. The initial version of the program was developed by R. Guigó et al. *(11)* in 1991 at the Molecular Biology Computer Research Resource (Dana Farber Cancer Institute, Harvard University). A completely new implementation of the system was written and released in 2000 at the IMIM in Barcelona, Spain *(12)*. Recently, a more powerful version of GeneID has been published, including new parameter configurations for a larger number of species *(13)*. Many other ab initio gene finding programs such as GENSCAN *(14)*, GENEMARK *(15)*, GRAIL *(16)*, or FGENESH *(17)* have also been published during the past years. GeneID, while having an accuracy comparable to the most of them *(18)*, is very efficient at handling large genomic sequences in terms of memory and speed. GeneID is able to analyze chromosome-size sequences in a few minutes on a standard workstation, and has a rich set of output options, which allow for a detailed analysis of gene features in genomic sequences. In fact, GeneID has been used as a component of the ab initio gene prediction pipeline in the genome annotation projects of *Dictyostelium discoideum (19)*, *Tetraodon nigroviridis (20)*, and *Paramecium tetraurelia (21)*. GeneID predictions precomputed in the genomes of human, mouse, and other species are served via DAS (Distributed Annotation System) through the UCSC genome browser *(8)* and the Ensembl site *(9)*. These predictions can also be found at the GeneID web site at http://genome.imim.es/software/geneid.

Despite the accuracy of gene annotations has been remarkably improved since the appearance of comparative genomics techniques (*see 22* for a recent review), current gene prediction tools are not yet powerful enough to precisely elucidate the structure of the genes from large genomic sequences. In addition, the definition of what a gene is has evolved in the past years according to novel biological evidence *(23)*. Thus, conventional gene identification techniques cannot deal with many classes of gene structures that do not fit into the classical definition of a gene such as alternative splicing *(24)*, pseudogenes, or selenoproteins *(25)*. These gene forms must be usually predicted apart with tools specifically designed for such purpose (e.g., *see 26* for an example of selenoprotein gene prediction pipeline using GeneID).

2. Program Usage

2.1. Informatic Resources

A computer with an internet connection is required to perform the example introduced throughout the chapter. Any internet browser (Firefox, IExplorer, Safari) can be used to access the genomic data and the gene prediction programs. There is no specific requirement for the computer operating system as we are using web servers to run the applications. The protocols that are explained in this chapter can be, however, performed locally in the computer as well. Local protocols are strongly recommended to automatically deal with great volumes of genomic information.

2.2. Genomes and Sequences

The genomic sequences used in this chapter can be freely retrieved from the UCSC genome browser *(8)*. Users are encouraged, however, to practice also with other similar resources such as ENSEMBL *(9)* or the NCBI genome viewer *(10)*. Different gene catalogues are available for *D. melanogaster*. Here, we have selected the Reference Sequence collection *(27)* and the FlyBase database *(28)*. The location in the UCSC genome browser of the two genomic sequences that are analyzed in the next section is (1) *D. melanogaster* genome (Apr. 2004 assembly: chr3R:16,118,988–16,127,097) and (2) *D. yakuba* genome (Nov. 2005 assembly: chr3R:4,894,105–4,902,627).

2.3. GeneID and Other Gene Prediction Programs

The program GeneID is designed following a simple hierarchical structure *(13)*: First, potential exon-defining signals (splice sites and start codons) are predicted and scored using position weight arrays derived from a subset of real signals *(12)*. Next, potential exons are constructed from these sites. The coding properties of such exons are evaluated running several Markov models, which

were calculated from a set of real exons and introns, on the hexamer composition of them *(12)*. Finally, the gene structure is assembled from the set of predicted exons, maximizing the sum of the scores of the selected exons *(29)*.

GeneID is distributed in two different forms, which are publicly available: a web server interface and a stand-alone Unix command-line program. In both cases, GeneID requires as input a file containing a DNA sequence in FASTA format and a parameter file. The parameter file contains the description of the probabilistic model on which the predictions are based and the gene model rules to assemble the final prediction *(13)*. Currently, there are parameter files for *Homo sapiens* (mammals), *Tetraodon nigroviridis* (*Fugu rubripes*), *Drosophila melanogaster* (other diptera species), *Caenorhabditis elegans, Dictyostelium discoideum, Solanum lycopersicum, Triticum aestivum, Oryza sativa, Arabidopsis thaliana,*and many others.

GeneID produces results in plain text which can then be saved into a file or submitted to another program (e.g., visualization tools). By default, the output consists of a series of genes predicted along the input sequence. Predicted genes are described as lists of potential coding exons. After the set of lines corresponding to the exons of each predicted gene, the amino acid sequence of the protein is printed in FASTA format. Each exon is defined by a start signal (start codon or acceptor site), an end signal (donor site or stop codon), the strand, and the frame. Each exon (as well as each signal) is assigned a score that indicates the reliability of the prediction. The score depends on the scores of the defining sites and on the nucleotide composition of the exon sequence. The score of a gene is the sum of the scores of its exons. In addition to the predicted genes, GeneID provides a number of options which allow the investigator to print an exhaustive list of all the sequence signals and exons predicted along the query sequence (most of which are not included in the final gene prediction).

GENSCAN *(14)* and GENEMARK *(15)* are other popular gene prediction programs, which are based on the application of hidden Markov models. In both cases, the genomic sequences are treated as language sentences that must be parsed into different exonic, intronic, and intergenic fragments according to a model or grammar of gene structure rules. These rules are specified as a set of nucleotide emission probabilities at different states (gene features) and the weighted transitions among them.

2.4. Other Bioinformatics Applications

Other applications that are used during the annotation pipeline introduced in this chapter are the sequence alignment programs BLAT *(30)* and CLUSTALW *(31)*, the program GENEWISE *(32)*, and the visualization tool GFF2PS *(33)*. The web tools used in this chapter are publicly available. However, a copy of most programs can be also downloaded into a local computer without additional cost (*see* **Note 1**).

3. Example

Here, we are going to use the gene *capicua* to introduce the classical strategies to find genes in recently assembled genomes for which no annotations are available yet. The gene *capicua* (*cic*,"head and tail" in catalan language, a palindromic number *(34)*) is a transcriptional repressor required for the specification of numerous cell types during embryonic development. *Capicua* is particularly relevant in the development of head and tail structures, dorsoventral patterning, and wing development *(35)*.

The genome of *D. melanogaster* was released in 1999 *(36)*. The current catalogue of genes of the fruit fly is available through the UCSC genome browser *(8)*. The gene *cic* is annotated on the chromosome 3R from *D. melanogaster*. We will use this annotation in such a genome to predict the location and the gene structure of *cic* in the genome of *D. yakuba*, whose assembly has been recently completed. First, we must detect the syntenic region in *D. yakuba* in which the gene is suspected to be present. Second, we will run GeneID to build an initial prediction of the gene. The predictions of other gene finding programs can be used to solidify the initial results. Finally, we will combine ab initio gene finding with homology searches (e.g., the protein product from *D. melanogaster*) to refine the resulting gene structure.

3.1. Characterize the Gene cic *Annotated in* Drosophila melanogaster

Before the annotation of the gene *cic* in *D. yakuba*, we should analyze the annotation of this gene in *D. melanogaster*. We must connect to the UCSC Genome Bioinformatics server at http://genome.ucsc.edu/. To access the fruit fly genome annotations, we can go to the link "Genome Browser" at the menu on the left (*see* **Note 2**). To get the annotation of the gene *cic*, select the clade Insect and the genome *D. melanogaster*. Now type the gene name *capicua* and submit the query. The UCSC genome browser can retrieve the genomic location of a gene using different alternative gene identifiers. For example, the values "capicua," "cic," "NM_080253" (RefSeq), and "FBgn0028386" (FlyBase) produce the same search results. You can find more information about the gene queries in the search web page. We must select now the link "cic at chr3R:16,118,988–16,127,097" (*see* **Note 3**), below the RefSeq genes collection to access the information. The Reference Sequence (RefSeq) collection is a high-quality nonredundant set of sequences, including genomic DNA, transcript (RNA), and protein products for human and many other organisms. RefSeq is maintained and curated at the NCBI *(27)*.

You are observing the annotation of the gene *cic* in the *D. melanogaster* genome. The information is displayed in different tracks. Each track corresponds to a different feature that has been

annotated on this region. For instance, the FlyBase and RefSeq tracks are showing the position of the exons of the gene *cic*. The visualization of each feature can be configured by the user. The options that are below the picture are divided into the following conceptual blocks: "Mapping and Sequencing," "Genes and Gene Prediction," "mRNA and EST," "Expression and Regulation," "Comparative Genomics," and "Variation and Repeats." A help page with additional information about each option is shown when the link with its name is selected. The interaction with the graphical annotations allows the user to configure the visualization options as well. You must activate only the following options (**Fig. 12.1**): Base Position, FlyBase Genes, RefSeq Genes, *D. melanogaster* mRNAs, Spliced ESTs, and Conservation.

Let us analyze the abundant information contained in this entry. We must click over the FlyBase Genes track to see the associated information: the gene *cic* is located at the strand (+) in the chromosome 3R, coordinates 16,118,988 and 16,127,097 (*see* **Note 4**). The size of the gene is 8,110 nucleotides, being constituted of 13 exons (*see* **Note 5**). We can see also the gene *cic* encodes a transcription factor (repressor) required for the specification of numerous cell types during embryonic development (body patterning). In addition, there is also available

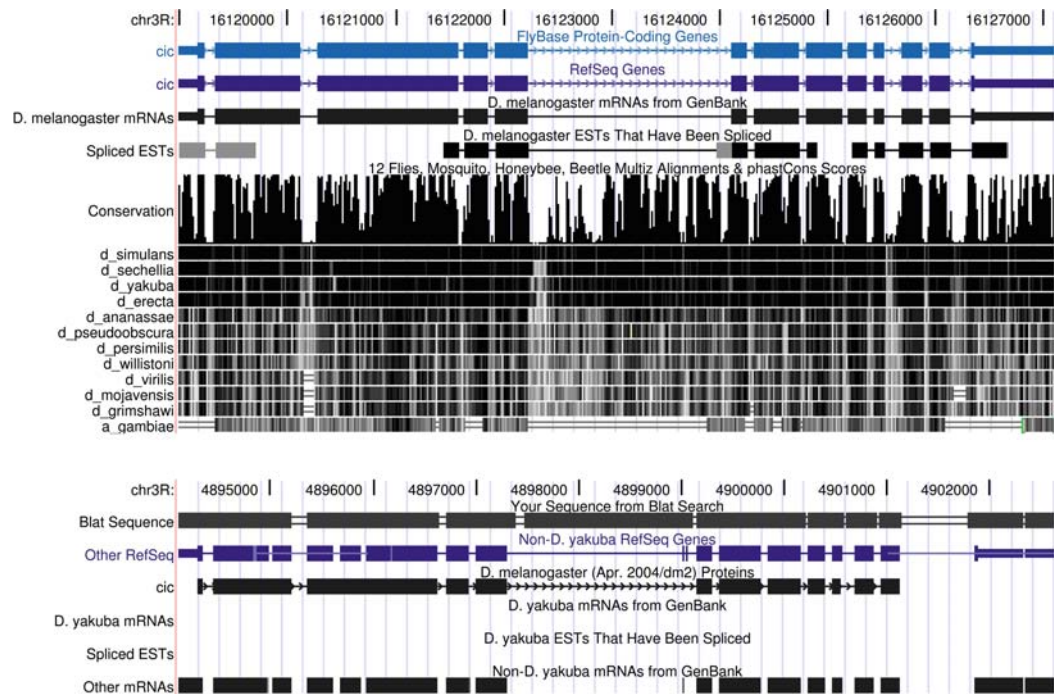


Fig.12.1. **Genomic annotation of the *capicua* gene** (*Top*) Current annotation of the gene *capicua* in *D. melanogaster* (*Bottom*) The syntenic region in *D. yakuba* in which the gene *capicua* is located. Both pictures have been obtained from the UCSC genome browser *(8)*.

information about the expression of this gene (Microarray Expression Data), protein features (Protein Domains and Structure), orthology (Orthologous genes in Other Species), and alleles (Phenotypes). You can have a look at the functional information too (Gene Ontology Annotations). We must go now to the RefSeq gene entry using the RefSeq Accession at the end of the web page (Other Names for This Gene).

The RefSeq identifier of *cic* is NM_080253 and the NCBI Entrez gene identifier is 53560. You can see the links to extract the protein and the mRNA sequences. The link "Genome Sequence" can be used to extract samples of variable lengths from the sequence of different components of the gene. This option can be extremely useful, for instance, to analyze the promoter region, the exon/intron boundary signals, the 3′UTR, or even the CDS of any gene that is annotated in this genome. There are other valuable resources that must be visited (*see* **Note 6**). Click over the NCBI Entrez gene link to explore the information available in Entrez for this gene.

Once we have finished, we can go back by clicking over the link "Position: chr3R:16,118,988–16,127,097" in the RefSeq entry to retrieve again the main picture of the gene annotations. The different tracks in the image are also important sources of information: click over the mRNA information track. You can see that two complete mRNAs (BT024235 and AJ252268) are supporting the annotation of this gene. The EST information can be explored as well. Expand the tracks to see the whole set of spliced ESTs. The majority of the exons are covered by at least two or more spliced ESTs (*see* **Note 7**). Finally, we can have a look at the alignment of the same syntenic region in different species. Comparative genomics is very useful to elucidate the position of functional regions in one genome for which no information is available yet. Functional elements (e.g., genes) can be then identified by using an informant genome for which a possible annotation already exists. In the conservation track, the peaks of the graphical lines corresponding to sequence alignments between the gene in *D. melanogaster* and the orthologs in multiple Drosophilas tend to be focused on the regions in which the real exons are annotated (*see* **Note 8**).

3.2. Searching the Syntenic Region that Contains the Gene cic *in* D. yakuba

Before running GeneID, we need to know the syntenic region from *D. yakuba* in which the gene should be annotated. We are going to use the program BLAT, which is part of the UCSC genome browser tools *(30)*, to study the alignments between both species. Basically, BLAT performs the alignment of highly similar regions between a query sequence and a precomputed index of the whole set of genomic sequences of other species. In particular, we are interested in finding the region of *D. yakuba* in which the gene from *D. melanogaster* can be successfully mapped.

First, we need to get the genomic sequence that contains the gene *cic* in the fruit fly before starting the BLAT search. We must connect to the UCSC Genome Bioinformatics server at http://genome.ucsc.edu/. To access the annotations of the gene *cic*, we can go to the link "Genome Browser" at the menu on the left. Then, we must select the clade Insect and the genome *D. melanogaster*. Finally, we must type the gene name *capicua* and submit the query. Go now to the link below the RefSeq genes collection: "cic at chr3R:16,118,988–16,127,097." The genomic region that is shown in the image can be retrieved in FASTA format using the link "DNA" in the menu on top (press inside the button "get DNA"). We will use this sequence to search for the syntenic region in *D. yakuba* with the program BLAT.

To run BLAT, you must go back to the UCSC homepage (open a new window). Click over the BLAT link (menu on Top) to access the input form. We need to set the genome *D. yakuba* in the corresponding box. Then, we can copy and paste the genomic sequence from *D. melanogaster* that was obtained previously. Submit the query to get the set of best hits between the sequence and the genome. We must click over the link "browser" of the first hit "3R + 4,894,105 4,902,627" to access that particular genomic fragment in *D. yakuba* (*see* **Note 9**). You are observing the available annotation of the putative syntenic region that contains the gene *cic* in *D. yakuba*(**Fig. 12.1**). The track "Blat sequence" displays the result of the alignment search. You should activate only the following options: Base Position, Other RefSeq, D. mel. Protein, *D. yakuba* mRNAs, Spliced ESTs, and Other mRNAs.

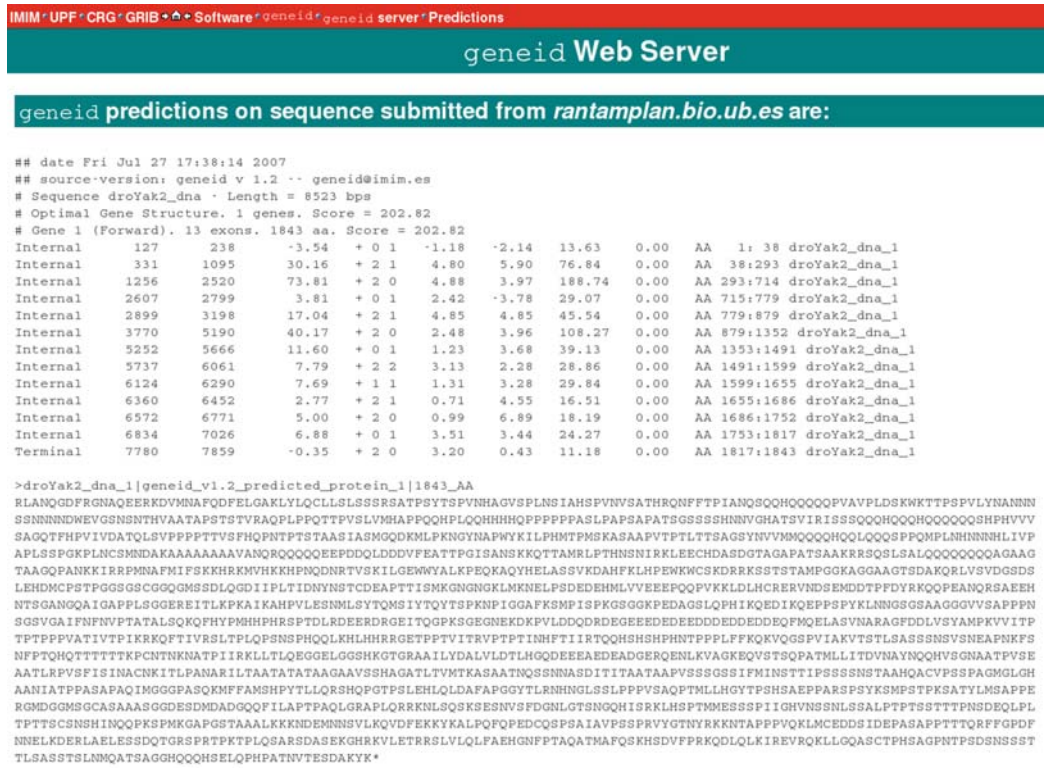
This genome is still being annotated: there are no available genes, mRNAs, or spliced ESTs from *D. yakuba* (empty tracks). However, we can extract useful information from the annotations of other species that were mapped in the same region. For instance, we can visually compare the alignment of the protein in *D. melanogaster* to the syntenic region. In practice, the protein sequence can be mostly mapped over the conserved fragments in this region (coding exons). Interestingly, the alignment of the RefSeq gene from *D. melanogaster* reveals the complete mRNA can be mapped over the conserved fragments as well (coding and UTR exons). In fact, the syntenic region detected by BLAT is very likely to be the correct one. Using the zoom out tool, we can see that the neighbor-mapped genes of *cic* are the same: CG10883, CG5060, cic, CG4367, CG4362 (*D. melanogaster* gene names). We can connect to the NCBI Entrez Gene at http://www.ncbi.nlm.nih.gov/ to verify that the gene has not yet been annotated in GenBank. Just select "Gene" at the Search box in the main page and type "yakuba & capicua" (*see* **Note 10**). No positive result is obtained. In consequence, according to the information we have gathered from different sources, we can conclude the gene *capicua* is not annotated in the *D. yakuba* genome assembly yet.

3.3. Ab Initio Gene Prediction in the Syntenic Region in D. yakuba

In the previous section, we have identified the region from *D. yakuba* in which we should probably find the gene *cic*. Here, we will use GeneID and two additional programs to characterize a preliminar gene structure (exons) of the gene. The sequence that was identified in the BLAT alignments can be easily retrieved again. Just connect to the UCSC Genome Bioinformatics server at http://genome.ucsc.edu/, click over the link "Genome Browser," select the clade Insect, the genome *D. yakuba*, and type the gene name *capicua*. We must select the link "cic at chr3R:4,894,105–4,902,627," below the RefSeq genes collection, to see the genomic region from *D. yakuba* where we are going to characterize the gene *cic*. You can use the button "DNA" to extract the genomic sequence.

We are going to obtain the initial gene prediction using the program GeneID. You can connect to the GeneID web server at http://genome.imim.es/geneid.html. To get the prediction, we must copy and paste the syntenic sequence from *D. yakuba* into the first box (Fasta sequence), select the organism *D. melanogaster* (*see* **Note 11**), set the "geneid" output format, and submit the query. Let us analyze the prediction (**Fig. 12.2**). GeneID identified one gene with 13 exons. The predicted gene is lacking of an initial exon. The length of the protein product is 1,843 amino acids. Apparently, most predicted exons are supported by a fragment of the protein from *D. melanogaster* (**Fig. 12.3**).

To accurately measure the quality of the prediction, we can perform the sequence alignment of the predicted protein product and the real protein with the program CLUSTALW *(31)*. First, we are going to recover the sequence of the real protein in fruit fly from the UCSC genome browser at http://genome.ucsc.edu/. Just search again the gene *cic* in *D. melanogaster* and click over the RefSeq track. The sequence of the protein product can be retrieved from the link "Predicted Protein" (a translation of the RefSeq transcript). We can now open the EBI CLUSTALW server at http://www.ebi.ac.uk/clustalw. We need to copy and paste the RefSeq protein and the protein predicted by GeneID into the CLUSTALW form. The global alignment of both sequences will be shown in a few seconds (*see* **Note 12**). Let us analyze the alignment. In general terms, most GeneID exons are supported by protein evidence. However, some regions of the protein are apparently not detected correctly (**Fig. 12.4**). The first exon predicted by GeneID contains the actual first exon in *D. melanogaster*, including a few additional nucleotides at the beginning. The end of the protein predicted by GeneID is longer than the real protein as well. This time, however, the actual last exon in *D. melanogaster* is not included into the prediction. In addition, there is a long internal exon in the prediction that is not matching the orthologous gene in *D. melanogaster* (**Table 12.1**).



IMIM · UPF · CRG · GRIB · Software · geneid · geneid server · Predictions

geneid Web Server

geneid predictions on sequence submitted from *rantamplan.bio.ub.es* are:

```
## date Fri Jul 27 17:38:14 2007
## source-version: geneid v 1.2 -- geneid@imim.es
# Sequence droYak2_dna · Length = 8523 bps
# Optimal Gene Structure. 1 genes. Score = 202.82
# Gene 1 (Forward). 13 exons. 1843 aa. Score = 202.82
Internal      127      238      -3.54    + 0 1    -1.18    -2.14    13.63    0.00    AA    1: 38 droYak2_dna_1
Internal      331     1095      30.16    + 2 1     4.80     5.90    76.84    0.00    AA   38:293 droYak2_dna_1
Internal     1256     2520      73.81    + 2 0     4.88     3.97   188.74    0.00    AA 293:714 droYak2_dna_1
Internal     2607     2799       3.81    + 0 1     2.42    -3.78    29.07    0.00    AA 715:779 droYak2_dna_1
Internal     2899     3198      17.04    + 2 1     4.85     4.85    45.54    0.00    AA 779:879 droYak2_dna_1
Internal     3770     5190      40.17    + 2 0     2.48     3.96   108.27    0.00    AA 879:1352 droYak2_dna_1
Internal     5252     5666      11.60    + 0 1     1.23     3.68    39.13    0.00    AA 1353:1491 droYak2_dna_1
Internal     5737     6061       7.79    + 2 2     3.13     2.28    28.86    0.00    AA 1491:1599 droYak2_dna_1
Internal     6124     6290       7.69    + 1 1     1.31     3.28    29.84    0.00    AA 1599:1655 droYak2_dna_1
Internal     6360     6452       2.77    + 2 1     0.71     4.55    16.51    0.00    AA 1655:1686 droYak2_dna_1
Internal     6572     6771       5.00    + 2 0     0.99     6.89    18.19    0.00    AA 1686:1752 droYak2_dna_1
Internal     6834     7026       6.88    + 0 1     3.51     3.44    24.27    0.00    AA 1753:1817 droYak2_dna_1
Terminal     7780     7859      -0.35    + 2 0     3.20     0.43    11.18    0.00    AA 1817:1843 droYak2_dna_1

>droYak2_dna_1|geneid_v1.2_predicted_protein_1|1843_AA
RLANQGDFRGNAQEERKDVMNAFQDFELGAKLYLQCLLSLSSSRSATPSYTSPVNHAGVSPLNSIAHSPVNVSATHRQNFFTPIANQSQQHQQQQQPVAVPLDSKWKTTPSPVLYNANNN
SSNNNNDWEVGSNSNTHVAATAPSTSTVRAQPLPPQTTPVSLVMHAPPQQHPLQQHHHHQPPPPPPASLPAPSAPATSGSSSSHNNVGHATSVIRISSSQQQHQQQHQQQQQQSHPHVVV
SAGQTFHPVIVDATQLSVPPPPTTVSFHQPNTPTSTAASIASMGQDKMLPKNGYNAPWYKILPHMTPMSKASAAPVTPTLTTSAGSYNVVMMQQQQHQQLQQQSPPQMPLNHNNNHLIVP
APLSSPGKPLNCSMNDAKAAAAAAAAVANQRQQQQQEEPDDQLDDDVFEATTPGISANSKKQTTAMRLPTHNSNIRKLEECHDASDGTAGAPATSAAKRRSQSLSALQQQQQQQQAGAAG
TAAGQPANKKIRRPMNAFMIFSKKHRKMVHKKHPNQDNRTVSKILGEWWYALKPEQKAQYHELASSVKDAHFKLHPEWKWCSKDRRKSSTSTAMPGGKAGGAAGTSDAKQRLVSVDGSDS
LEHDMCPSTPGGSGSCGGQGMSSDLQGDIIPLTIDNYNSTCDEAPTTISMKGNGNGKLMKNELPSDEDEHMLVVEEEPQQPVKKLDLHCRERVNDSEMDDTPFDYRKQQPEANQRSAEEH
NTSGANGQAIGAPPLSGGEREITLKPKAIKAHPVLESNMLSYTQMSIYTQYTSPKNPIGGAFKSMPISPKGSGGKPEDAGSLQPHIKQEDIKQEPPSPYKLNNGSGSAAGGGVVSAPPPN
SGSVGAIFNFNVPTATALSQKQFHYPMHHPHRSPTDLRDEERDRGEITQGPKSGEGNEKDKPVLDDQDRDEGEEEDEDEEDDDEDDEDDEQFMQELASVNARAGFDDLVSYAMPKVVITP
TPTPPPVATIVTPIKRKQFTIVRSLTPLQPSNSPHQQLKHLHHRRGETPPTVITRVPTPTINHFTIIRTQQHSHSHPHNTPPPLFFKQKVQGSPVIAKVTSTLSASSSNSVSNEAPNKFS
NFPTQHQTTTTTTKPCNTNKNATPIIRKLLTLQEGGELGGSHKGTGRAAILYDALVLDTLHGQDEEEAEDEADGERQENLKVAGKEQVSTSQPATMLLITDVNAYNQQHVSGNAATPVSE
AATLRPVSFISINACNKITLPANARILTAATATATAAGAAVSSHAGATLTVMTKASAATNQSSNNASDITITAATAAPVSSSGSSIFMINSTTIPSSSSNSTAAHQACVPSSPAGMGLGH
AANIATPPASAPAQIMGGGPASQKMFFAMSHPYTLLQRSHQPGTPSLEHLQLDAFAPGGYTLRNHNGLSSLPPPVSAQPTMLLHGYTPSHSAEPPARSPSYKSMPSTPKSATYLMSAPPE
RGMDGGMSGCASAAASGGDESDMDADGQQFILAPTPAQLGRAPLQRRKNLSQSKSESNVSFDGNLGTSNGQHISRKLHSPTMMESSSPIIGHVNSSNLSSALPTPTSSTTTPNSDEQLPL
TPTTSCSNSHINQQPKSPMKGAPGSTAAALKKKNDEMNNSVLKQVDFEKKYKALPQFQPEDCQSPSAIAVPSSPRVYGTNYRKKNTAPPPVQKLMCEDDSIDEPASAPPTTTQRFFGPDF
NNELKDERLAELESSDQTGRSPRTPKTPLQSARSDASEKGHRKVLETRRSLVLQLFAEHGNFPTAQATMAFQSKHSDVFPRKQDLQLKIREVRQKLLGQASCTPHSAGPNTPSDSNSSST
TLSASSTSLNMQATSAGGHQQQHSELQPHPATNVTESDAKYK*
```

Fig.12.2. **GeneID prediction of the gene *cic*** GeneID *(13)* identified one gene constituted of 13 exons on the syntenic region of *D. yakuba*. The symbol "#" is used to denote such a line of the prediction contains information about gene and sequence general features. For each predicted exon the following information is displayed: exon type, begin and end coordinates, total score, strand, frame and remainder, *left* splice site score, *right* splice site score, coding potential score, homology information, amino acid coordinates, and gene internal identifier. The protein product is displayed below the gene prediction.

We can obtain additional clues that support the GeneID prediction using other gene finding programs. We are going to perform the gene prediction on the sequence from *D. yakuba* using the programs GENSCAN and GENEMARK. The GENSCAN web server is accessible at http://genes.mit.edu/GENSCAN.html. To obtain the predicted gene, we must copy and paste the genomic region from *D. yakuba* and run GENSCAN. Let us analyze the prediction: GENSCAN identified one gene with 11 exons (**Table 12.1**). The predicted gene is lacking of an initial exon and a terminal exon. The length of the protein product is 1,770 amino acids. Many predicted exons are supported by a fragment of the protein from *D. melanogaster* **(Fig. 12.3)**. The GENEMARK web server can be used at http://exon.gatech.edu/genemark/eukhmm.cgi. Just copy and paste the genomic region from *D. yakuba*, select the species *D. melanogaster*, and press the button "Start GeneMark.hmm." GENEMARK identified one

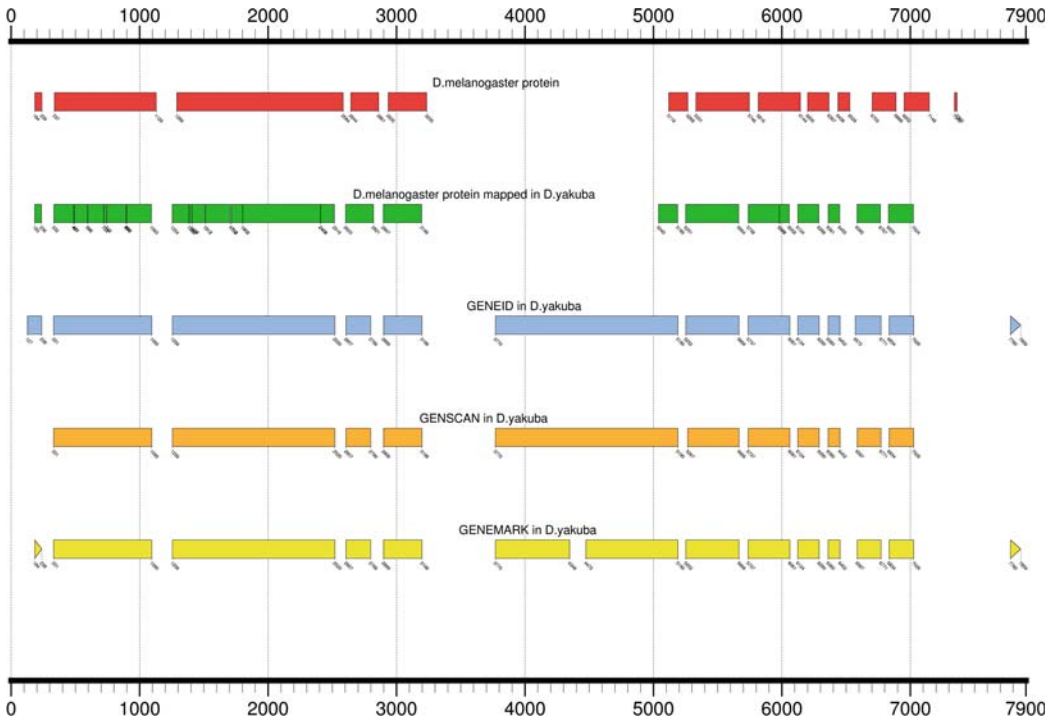


Fig.12.3. **Comparison between the protein and the gene predictions**
The product of the gene *capicua* in *D. melanogaster* has been previously mapped on the genome of *D. yakuba*. The graphical representation of the annotations has been obtained using the program GFF2PS *(33)*.

gene with 14 exons (**Table 12.1**). The length of the protein product is 1,776 amino acids. Many predicted exons are supported by a fragment of the protein from *D. melanogaster* (**Fig. 12.3**).

We can compare the predictions obtained with these programs (exon coordinates) to study the common exons and the differences in the results (**Table 12.1**). In addition, we can produce graphical

```
NP_524992 ..................MNAFQDFELGAKLYLQCLLSLSSSRSATPSYTSPVN   36
geneid    RLANQGDFRGNAQEERKDVMNAFQDFELGAKLYLQCLLSLSSSRSATPSYTSPVN   55

NP_524992 HAGASPLNAIAHSPVNVSATHRQNFFTPIANQSQQ.QQQQQPVAV   80
geneid    HAGVSPLNSIAHSPVNVSATHRQNFFTPIANQSQQHQQQQQPVAV  100

NP_524992 AFQSKHSDVFPRKQDLQLKIREVRQKLLGQASCTPHSAGPNTPSDSNSSSTTLSA 1384
geneid    AFQSKHSDVFPRKQDLQLKIREVRQKLLGQASCTPHSAGPNTPSDSNSSSTTLSA 1804

NP_524992 SSTSLNMQTTSAADVFQYY.................. 1403
geneid    SSTSLNMQATSAGGHQQQHSELQPHPATNVTESDAKYK 1842
```

Fig.12.4. **Sequence alignment between the protein and the predicted gene**
The prediction was obtained using the program GENEID *(13)*. The alignment between the prediction and the protein (NP_524992) from *D. melanogaster* was performed with the program CLUSTALW *(31)*. Only the first and the last exon are shown here, above and below the line, respectively.

Table 12.1
Specificity of the predictions in *D. yakuba* (*see* Note 16)

EXON	$GENEID_{yakuba}$	$GENSCAN_{yakuba}$	$GENEMARK_{yakuba}$	PROTEIN evidence
E1	127–238		184–238	**OK**
E2	331–1,095	331–1,095	331–1,095	**OK**
E3	1,256–2,520	1,256–2,520	1,256–2,520	**OK**
E4	2,607–2,799	2,607–2,799	2,607–2,799	**OK**
E5	2,899–3,198	2,899–3,198	2,899–3,198	**OK**
E6			3,770–4,348	NO
E7			4,475–5,190	PARTIAL
E8	3,770–5,190	3,770–5,190		PARTIAL
E9	5,252–5,666	5,267–5,666	5,252–5,666	**OK**
E10	5,737–6,061	5,737–6,061	5,737–6,061	**OK**
E11	6,124–6,290	6,124–6,290	6,124–6,290	**OK**
E12	6,360–6,452	6,360–6,452	6,360–6,452	**OK**
E13	6,572–6,771	6,587–6,771	6,587–6,771	**OK**
E14	6,834–7,026	6,834–7,026	6,834–7,026	**OK**
E15	7,780–7,859		7,780–7,859	NO

The coordinates and the protein supporting information are provided for each exon that was predicted by GENEID *(13)*, GENSCAN *(14)*, and GENEMARK *(15)*. The exons supported by fragments of the protein in *D. melanogaster*were obtained from the alignment between both sequences.

representations of the genomic annotations. Here, the program GFF2PS *(33)* has been used to produce the graphical annotations in this chapter. However, each gene finding program typically creates its own graphical annotation using bitmaps (JPG, PNG, TIFF) or vector graphical formats (PDF, PostScript). If we analyze the graphical representation of the predictions in **Fig. 12.3**, we can observe that most exons in the protein from *D. melanogaster* are correctly identified by the gene finding programs. However, there are several inconsistencies as well (**Table 12.1**). The initial exon was correctly predicted by GENEMARK and partially detected by GeneID, while it was not found by GENESCAN (*see* **Note 13**). In addition, the internal exons between positions 4,000 and 5,000 included in the three predictions are not supported by any fragment of the protein evidence (**Fig. 12.3**). Finally, the last exon of the gene was not found by any program (*see* **Note 14**).

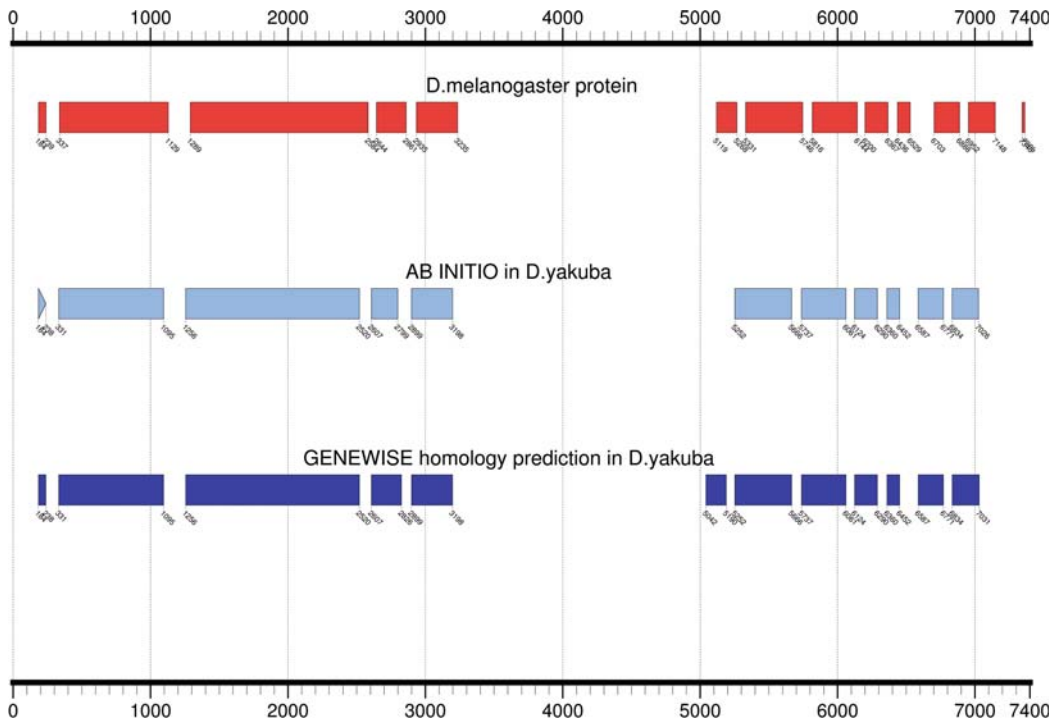


Fig.12.5. **Comparison between ab initio prediction, homology prediction and the real protein**
The original protein from *D. melanogaster* is shown here. The graphical representation of the annotations has been obtained using the program GFF2PS *(33)*.

Gathering all of this information, we can reconstruct an initial prediction of the gene *cic* in *D. yakuba* (**Fig. 12.5**). The resulting gene is constituted of the initial exon predicted by GeneID and GENEMARK, and the common internal exons that are conserved in the three predictions (except the exon that was not supported by protein information). The terminal exon predicted by GeneID and GENEMARK is not included initially into this conservative annotation because of the lack of protein support. Special attention in a posterior study should be paid to this exon, however, because of its short length and the fact that it was detected by two independent programs. In any case, the ab initio prediction of the gene *cic* in *D. yakuba* was able to recognize 12 out of 13 exons that are constituting the protein in *D. melanogaster* (**Table 12.2**).

3.4. Using Homology Information: Spliced Alignment

In some situations, we might be interested in the automatical incorporation of protein homology information into the gene prediction. For instance, the program GENEWISE *(32)* allows us to perform the alignment between ab initio signal predictions with protein homology information from different species. Thus, we can integrate exon splicing predictions in the syntenic region from *D. yakuba* with the protein product of the gene *cic* in *D. melanogaster*.

Table 12.2
Sensitivity of the predictions in *D. yakuba* (*see* Note 16)

EXON	POSITION1	POSITION2	$GENEID_{yakuba}$	$GENSCAN_{yakuba}$	$GENEMARK_{yakuba}$
E1	184	239	**OK**	NO	**OK**
E2	337	1,129	**OK**	**OK**	**OK**
E3	1,289	2,584	**OK**	**OK**	**OK**
E4	2,644	2,861	**OK**	**OK**	**OK**
E5	2,935	3,235	**OK**	**OK**	**OK**
E6	5,119	5,268	**OK**	**OK**	**OK**
E7	5,331	5,746	**OK**	**OK**	**OK**
E8	5,816	6,144	**OK**	**OK**	**OK**
E9	6,200	6,367	**OK**	**OK**	**OK**
E10	6,436	6,529	**OK**	**OK**	**OK**
E11	6,703	6,888	**OK**	**OK**	**OK**
E12	6,952	7,148	**OK**	**OK**	**OK**
E13	7,343	7,363	NO	NO	NO

We have used the fragments of the protein from *D. melanogaster*, which possibly constitute the exons of the gene, to evaluate the accuracy of the predictions by GENEID, GENSCAN and GENEMARK. The protein was aligned previously to the syntenic region in *D. yakuba* in which the gene should be identified.

We can connect to the GENEWISE web server at http://www.ebi.ac.uk/Wise2/ (*see* **Note 15**). First, we need to get from the UCSC genome browser the translated protein from the RefSeq track in *D. melanogaster*, and the genomic syntenic region from *D. yakuba* again. Then, both sequences must be copied into their respective boxes in the GENEWISE form to run the program. Let us explore the predictions: the alignment between the collection of predicted splice sites and the real protein is shown vertically – each column contains the alignment between the amino acids in the input protein and the corresponding codons in the input genomic region; introns are represented as gaps within such an alignment (**Fig. 12.6**). GENEWISE alignment is very accurate in most exons. Despite the ab initio prediction obtained in the previous section was not significantly improved, GENEWISE identified more accurately the location of the internal exon around the position 5,000, which was wrongly predicted before. On the contrary, the actual terminal exon was not included into the final alignment (**Fig. 12.5**).

```
NP_524992       1264 DFNNELK                              LESSDQTGRSPRTPKT
                     DFNNELK                              LESSDQTGRSPRTPKT
                     DFNNELK           E:E[gaa]           LESSDQTGRSPRTPKT
droYak2_dna     6431 gtaagcaGGTAAGCT  Intron 10  CAGAAcgatgcagctcaacaa
                     ataaata <1-----[6453 : 6586]-1>  tagcaacggccgccac
                     ctccaag                          ggctcagacacggaga
```

Fig.12.6. **Spliced alignment**
The alignment between the protein in *D. melanogaster* and the syntenic region in *D. yakuba* was performed using the program GENEWISE *(32)*. Only a fragment of the output containing one internal exon is shown here.

4. Conclusions

Gene prediction is not an easy problem. However, the complexity can be notably reduced when the proper sources of information are integrated in the analysis. Here, we have combined ab initio gene predictions (such as those obtained by GeneID) with protein annotations to retrieve a feasible gene structure in a genome that has not been annotated yet. The integration of sequence alignment and gene predictions will probably produce positive results in many genes. However, other more elaborated approaches are certainly necessary to tackle alternative gene forms, in which the analysis of individual predicted splice sites and exons might be crucial. In practice, the use of several gene prediction programs can provide interesting results. We have only focused on the study of the exons that were conserved in most predictions, but those that were not conserved might deserve special attention as well. In addition, we have used the protein from *D. melanogaster* as a reference to define the correctness of the predictions, while the actual protein in *D. yakuba* could be slightly different. In any case, the core of the protein was successfully identified in the genome of *D. yakuba*, which confirms the usefulness of the protocol presented in this case study.

Last exon of the gene *cic* from *D. melanogaster* is constituted of only 21 coding nucleotides while the rest is part of the 3′UTR region (**Fig. 12.1**). As a consequence, the ab initio gene prediction programs, such as GENEID, GENSCAN, or GENEMARK, and the homology-based methods, such as GENEWISE, are failing to detect the last seven amino acids of the protein in *D. melanogaster*. With this information, we strongly recommend to start a comparative genomics analysis with the predictions of the same gene in other species of drosophila to accurately establish the coordinates and the conservation of such an exon. For example, the multiple sequence alignment of the syntenic regions and the predictions of comparative gene finding tools *(37, 38)* could reveal the exact position, helping to refine the final annotation in *D. yakuba* as well.

5. Notes

1. We encourage the readers to write and use open source software as much as possible to speed up the improvement and the research on these applications. GeneID, for example, is release under the GNU-GPL license.
2. We are accessing the sequence and the annotation of the genomes through the web interface. However, genomic data can be also downloaded into the user's computer using the Downloads link in the same menu.
3. Genomic location (coordinates) of a gene feature can be different from an older assembly version to the next one, as the sequence is refined and the remaining gaps can be determined.
4. Genomic sequences are encoded in the databases as strings of nucleotides with an arbitrary orientation (5′ to 3′ DNA sequences), which receives the name of forward strand or (+). The reverse and complement sequence is named reverse or (−)
5. The 13 exons of the gene *cic* are coding exons (CDS). The first and the last exons, however, are also containing the 5′UTR and the 3′UTR fragments of the gene (such exons are displayed in a slightly different format in the graphical map).
6. The different annotation databases usually provide cross-links to compare the information that is available for each gene at each database. Thus, the user can visit the annotations in FlyBase or RefSeq from the Entrez gene or vice versa.
7. Expressed sequence tags (ESTs) are short fragments of the mRNA that have been sequenced. Those ESTs that contain at least one intronic connection are called "Spliced ESTs." Spliced ESTs are preferred to improve ab initio predictions as they provide strong evidence supporting the Donor/Acceptor splice site pair for a given intron.
8. Nowadays, there is an international on-going project to assemble, to align, and to annotate the genome of 12 drosophila species (AAA project: http://rana.lbl.gov/drosophila/).
9. The BLAT hits are evaluated using two different aspects: the length of the match between the query sequence from *D. melanogaster* and the fragment of the *D. yakuba* genome, and the similarity of such an alignment. By default, the Search Results are sorted by a combination of both values (Score). In this particular case, the first hit is covering the complete query sequence (8,110 nucleotides) with a high similarity (92.7% of identities). In fact, the score of such a hit is superior (6,325) in comparison to the rest.

10. If you just type "yakuba" you will retrieve the genes from *D. yakuba* annotated in GenBank. On the contrary, if you just type "capicua" you will retrieve the list of possible orthologs of this gene. The combination of both searches is indicated with the symbol "&" (AND). The same search can be performed in other databases from NCBI Entrez (protein, nucleotides), obtaining negative results as well. Here, we are assuming the gene name in *D. melanogaster* should be preserved in the available *D. yakuba* annotations.
11. Gene prediction programs are usually trained to take advantage of the specific features of genes in different organisms. Thus, customized parameter files for each group of related species are typically available (e.g., mammals, vertebrates, insects, plants).
12. There are basically two different strategies to align two biological sequences: (a) global alignment for sequences with similar content and length; (b) local alignment to detect short fragments conserved within two sequences that are quite different (*see 39* for a review). Here, we are interested in the alignment of two proteins that should be very similar (global alignment)
13. Initial exons are delimited by the exon-defining signals Start codon and Donor splice site. Start codons show a relatively low degree of conservation in their sequence among genes of the same species. The first CDS exon is very short in many cases as well (even containing a single nucleotide). These exons are, therefore, usually difficult to be predicted accurately because of both facts.
14. The terminal exon of the gene *cic* is extremely short, so that the protein-coding content cannot be properly evaluated by the Markov models of the gene finding programs. Thus, it is not surprising that the terminal exon in this particular case is wrongly predicted.
15. GENEWISE processing (spliced alignment) is constituted of a first step to predict splice sites and a second step in which the optimal alignment between the protein and the best splice sites is computed. The spliced alignment is a quite intensive procedure in terms of computational time and memory usage. Therefore, GENEWISE is recommended to be used only in short genomic regions in which the existence of the orthologous gene (protein) has been previously established.
16. The accuracy of gene predictions is usually measured in terms of sensitivity and specificity. At the exon level, sensitivity is defined as the proportion of real exons that are detected in the predictions, while specificity is defined as the proportion of predicted exons that are supported by the real annotations.

References

1. Blanco, E., and R. Guigó (2005) Predictive methods using DNA sequences, in *Bioinformatics : A Practical Guide to the Analysis of Genes and Proteins* (Baxevanis, A.D. and Ouellette, B.F.F. Eds). Wiley-Interscience: Hoboken, NJ, p. xviii, 540 p.
2. ENCODE Project Consortium (2007) Identification and analysis of functional elements in 1% of the human genome by the ENCODE pilot project. *Nature* **447**(7146), 799–816.
3. Zhang, M. Q. (2002) Computational prediction of eukaryotic protein-coding genes. *Nat Rev Genet* **3**(9), 698–709.
4. Venter, J. C., et al. (2001) The sequence of the human genome. *Science* **291**(5507), 1304–51.
5. Nagaraj, S. H., Gasser, R. B., and Ranganathan, S. (2007) A hitchhiker's guide to expressed sequence tag (EST) analysis. *Brief Bioinform* **8**(1), 6–21.
6. Stanke, M., Tzvetkova, A., and Morgenstern, B. (2006) AUGUSTUS at EGASP: using EST, protein and genomic alignments for improved gene prediction in the human genome. *Genome Biol* **7 Suppl 1**, S11 1–8.
7. Allen, J. E., and Salzberg, S. L. (2005) JIGSAW: integration of multiple sources of evidence for gene prediction. *Bioinformatics* **21**(18), 3596–603.
8. Kuhn, R. M., et al. (2007) The UCSC genome browser database: update 2007. *Nucleic Acids Res* **35**(Database issue), D668–73.
9. Hubbard, T. J., et al. (2007) Ensembl 2007. *Nucleic Acids Res* **35**(Database issue), D610–7.
10. Wheeler, D. L., et al. (2007) Database resources of the National Center for Biotechnology Information. *Nucleic Acids Res* **35**(Database issue), D5–12.
11. Guigo, R., et al. (1992) Prediction of gene structure. *J Mol Biol* **226**(1), 141–57.
12. Parra, G., Blanco, E., and Guigo, R. (2000) GeneID in *Drosophila*. *Genome Res* **10**(4), 511–5.
13. Blanco, E., Parra, G., and Guigó, R. (2007) Using geneid to identify genes *in Current Protocols in Bioinformatics* (Baxevanis, A. D. et al., Eds). John Wiley & Sons: New York, p. 1–28 (Unit 4.3).
14. Burge, C., and Karlin, S. (1997) Prediction of complete gene structures in human genomic DNA. *J Mol Biol* **268**(1), 78–94.
15. Besemer, J., and Borodovsky, M. (2005) GeneMark: web software for gene finding in prokaryotes, eukaryotes and viruses. *Nucleic Acids Res* **33**(Web Server issue), W451–4.
16. Uberbacher, E. C., and Mural, R. J. (1991) Locating protein-coding regions in human DNA sequences by a multiple sensor-neural network approach. *Proc Natl Acad Sci USA* **88**(24), 11261–5.
17. Salamov, A. A., and Solovyev, V. V. (2000) Ab initio gene finding in Drosophila genomic DNA. *Genome Res* **10**(4), 516–22.
18. Reese, M. G., et al. (2000) Genome annotation assessment in Drosophila melanogaster. *Genome Res* **10**(4), 483–501.
19. Glockner, G., et al. (2002) Sequence and analysis of chromosome 2 of Dictyostelium discoideum. *Nature* **418**(6893), 79–85.
20. Jaillon, O., et al. (2004) Genome duplication in the teleost fish Tetraodon nigroviridis reveals the early vertebrate protokaryotype. *Nature* **431**(7011), 946–57.
21. Aury, J. M., et al. (2006) Global trends of whole-genome duplications revealed by the ciliate Paramecium tetraurelia. *Nature* **444**(7116), 171–8.
22. Guigo, R., et al. (2006) EGASP: the human ENCODE Genome Annotation Assessment Project. *Genome Biol* **7 Suppl 1**, S2 1–31.
23. Gingeras, T. R. (2007) Origin of phenotypes: genes and transcripts. *Genome Res* **17**(6), 682–90.
24. Ladd, A. N., and Cooper, T. A. (2002) Finding signals that regulate alternative splicing in the post-genomic era. *Genome Biol* **3**(11), reviews0008.
25. Low, S. C., and Berry, M. J. (1996) Knowing when not to stop: selenocysteine incorporation in eukaryotes. *Trends Biochem Sci* **21**(6), 203–8.
26. Castellano, S., et al. (2004) Reconsidering the evolution of eukaryotic selenoproteins: a novel nonmammalian family with scattered phylogenetic distribution. *EMBO Rep* **5**(1), 71–7.
27. Pruitt, K. D., Tatusova, T., and Maglott, D. R. (2007) NCBI reference sequences (RefSeq): a curated non-redundant sequence database of genomes, transcripts and proteins. *Nucleic Acids Res* **35**(Database issue), D61–5.
28. Crosby, M. A., et al. (2007) FlyBase: genomes by the dozen. *Nucleic Acids Res* **35**(Database issue), D486–91.

29. Guigo, R. (1998) Assembling genes from predicted exons in linear time with dynamic programming. *J Comput Biol* **5**(4), 681–702.
30. Kent, W. J. (2002) BLAT – the BLAST-like alignment tool. *Genome Res* **12**(4), 656–64.
31. Thompson, J. D., Higgins, D. G., and Gibson, T. J. (1994) CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. *Nucleic Acids Res* **22**(22), 4673–80.
32. Birney, E., Clamp, M., and Durbin, R. (2004) GeneWise and Genomewise. *Genome Res* **14**(5), 988–95.
33. Abril, J. F., and Guigo, R. (2000) gff2ps: visualizing genomic annotations. *Bioinformatics* **16**(8), 743–4.
34. Fabra, P., and Miracle, J. (1983) Diccionari general de la Ilengua catalana. (17a ed). EDHASA editorial: Barcelona, 1786 p.
35. Jimenez, G., et al. (2000) Relief of gene repression by torso RTK signaling: role of capicua in Drosophila terminal and dorsoventral patterning. *Genes Dev* **14**(2), 224–31.
36. Adams, M. D., et al. (2000) The genome sequence of Drosophila melanogaster. *Science* **287**(5461), 2185–95.
37. Parra, G., et al. (2003) Comparative gene prediction in human and mouse. *Genome Res* **13**(1), 108–17.
38. Wang, M., Buhler, J., and Brent, M. R. (2003) The effects of evolutionary distance on TWINSCAN, an algorithm for pair-wise comparative gene prediction. *Cold Spring Harb Symp Quant Biol* **68**, 125–30.
39. Batzoglou, S. (2005) The many faces of sequence alignment. *Brief Bioinform* **6**(1), 6–22.

Chapter 13

Promoter Analysis: Gene Regulatory Motif Identification with A-GLAM

Leonardo Mariño-Ramírez, Kannan Tharakaraman, John L. Spouge, and David Landsman

Abstract

Reliable detection of *cis*-regulatory elements in promoter regions is a difficult and unsolved problem in computational biology. The intricacy of transcriptional regulation in higher eukaryotes, primarily in metazoans, could be a major driving force of organismal complexity. Eukaryotic genome annotations have improved greatly due to large-scale characterization of full-length cDNAs, transcriptional start sites (TSSs), and comparative genomics. Regulatory elements are identified in promoter regions using a variety of enumerative or alignment-based methods. Here we present a survey of recent computational methods for eukaryotic promoter analysis and describe the use of an alignment-based method implemented in the A-GLAM program.

Key Words: Promoter regions, transcription factor binding sites, enumerative methods, promoter comparison.

1. Introduction

The establishment and maintenance of temporal and spatial patterns of gene expression are achieved primarily by transcription regulation. Additionally, the precise control of timing and location of gene expression depends on the interaction between transcription factors and *cis*-acting sequence elements in promoter regions. Transcription factors can induce or repress gene expression upon binding of their cognate sequence element in the DNA. The discovery and categorization of the entire collection of transcription factor-binding sites (TFBSs) of an organism are among the greatest challenges in computational biology *(1)*. Large-scale efforts involving genome mapping and identification of TFBS in

David Posada (ed.), *Bioinformatics for DNA Sequence Analysis,* Methods in Molecular Biology 537

DOI 10.1007/978-1-59745-251-9_13

lower eukaryotes, such as the yeast *Saccharomyces cerevisiae*, have been successful *(2)*. In contrast, similar efforts in vertebrates have proven difficult due to the presence of repetitive elements and an increased regulatory complexity *(3–5)*.

The accurate prediction and identification of regulatory elements in higher eukaryotes remains a challenge for computational biology, despite recent progress in the development or improvement of different algorithms *(6–19)*. Different strategies for motif recognition have been benchmarked to compare their performance *(20)*. Typically, computational methods for identifying *cis*-regulatory elements in promoter sequences fall into two classes, enumerative and alignment techniques *(21)*. We have developed algorithms that use enumerative approaches to identify *cis*-regulatory elements statistically significant over-represented in promoter regions *(22)*. Subsequently, we developed an algorithm that combines both enumeration and alignment techniques to identify statistically significant *cis*-regulatory elements positionally clustered relative to a specific genomic landmark *(23,24)*.

Promoter identification is the first step in the computational analysis that leads to the prediction of regulatory elements. In lower Eukaryotes this is a rather simple problem due to a relative high gene density with respect to the genome size. The yeast *Saccharomyces cerevisiae* has ~70% of its genome coding for proteins and its intergenic regions are fairly short (~440 bp in length) *(25)*. In contrast, the human genome has a relative low gene density, with ~3% of the genome coding for proteins *(26)*; this poses significant challenges for the identification of both the promoter and its regulatory elements. Despite the complexity of gene expression regulation in higher Eukaryotes *(27)*, we now have a number of experimental and computational resources that can assist in the delineation of mammalian promoter regions. The experimental resources include full-length cDNA collections *(28)* and transcriptional start sites (TSS) *(29)*. Additionally, complementary computational resources include the database of transcriptional start sites (DBTSS) *(30)* and promoter identification services *(31–33)*. Many regulatory elements are located in the proximal promoter region (PPR) located a few hundred bases upstream the TSS *(22)* and the PPR can be generally defined by its low transposable element content *(34)*.

The computational methods for the prediction and identification of transcription factor binding sites can be divided in two broad categories: algorithms for de novo identification and algorithms that recognize elements using prior knowledge of the elements. Enumerative and alignment methods form part of the de novo algorithms. Enumerative algorithms use exhaustive methods to examine exact DNA words of a fixed length to rank them according to a specific function that determine over-representation relative to a background distribution. An enumerative

method that estimates *p*-values with the standard normal approximation associated with *z*-scores *(22)* has been successfully applied for the identification of regulatory elements in higher Eukaryotes *(35)*. Other enumerative methods include Weeder *(16, 17)*, oligonucleotide frequency analysis *(36)*, and QuickScore *(14)*.

Alignment methods aim to identify functional elements by a multiple local alignment of all sequences of interest. The most popular algorithms in this category use an optimization procedure based in probabilistic sequence models to find statistically significant motifs; these include Gibbs sampling *(37)* or expectation maximization *(11)*. Approaches that use a combination of enumerative and alignment methods have shown a significant improvement in the identification of regulatory elements in promoter sequences *(23, 24)*.

Algorithms that use prior knowledge of known motifs often use position frequency matrices (PFMs) that contain the number of observed nucleotides at each position *(38)*. Methods that assess statistical over-representation of known motifs in a set of sequences have been particularly successful *(9)*. Additionally, motif scores determined by over-representation can be used as a proxy to perform promoter comparisons *(39)*.

2. Program Usage

2.1. The A-GLAM Algorithm

The A-GLAM software package uses a Gibbs sampling algorithm to identify functional motifs in a set of sequences. Gibbs sampling, also known as successive substitution sampling, is a well-known Markov-chain Monte Carlo procedure for discovering sequence motifs *(37)*. In brief, A-GLAM takes a set of sequences in FASTA format as input. The Gibbs sampling step in A-GLAM uses simulated annealing to maximize an "overall score," corresponding to a Bayesian marginal log-odds score. The overall score is given by

$$s = \sum_{i=1}^{w} \left(\log_2 \frac{(a-1)!}{(c+a-1)!} + \sum_{(j)} \left\{ \log_2 \left[\frac{(c_{ij}+a_j-1)!}{(a_j-1)!} \right] - c_{ij} \log_2 p_j \right\} \right) \quad (1)$$

In equation (1), $m! = m(m-1)\ldots 1$ denotes a factorial; a_j, the pseudo-counts for nucleic acid j in each position; $a = a_1 + a_2 + a_3 + a_4$, the total pseudo-counts in each position; c_{ij}, the count of nucleic acid j in position i; and $c = c_{i1} + c_{i2} + c_{i3} + c_{i4}$, the total number of aligned windows, which is independent of the position i. The underlying principle behind the overall score s in A-GLAM is explained in detail elsewhere *(23)*.

The annealing maximization is initialized when A-GLAM places a single window of arbitrary size and position at every sequence, generating a gapless multiple alignment of the windowed subsequences. The program then proceeds through a series of iterations; on each iteration step, A-GLAM proposes a set of adjustments to the alignment. The proposal step is either a repositioning step or a resizing step. In a repositioning step, a single sequence is chosen uniformly at random from the alignment; and the set of adjustments include all possible positions in the sequence where the alignment window would fit without overhanging the ends of the sequence. In a resizing step, either the right or the left end of the alignment window is selected; and the set of proposed adjustments includes expanding or contracting the corresponding end of all alignment windows by one position at a time. Each adjustment leads to a different value of the overall score s. Then, A-GLAM accepts one of the adjustments randomly, with probability proportional to $\exp(s/T)$. A-GLAM may even exclude a sequence if doing so would improve alignment quality. The temperature T is gradually lowered to $T = 0$, with the intent of finding the gapless multiple alignments of the windows maximizing s. The maximization implicitly determines the final window size. The randomness in the algorithm helps it avoid local maxima and find the global maximum of s. However, due to the stochastic nature of the procedure, finding the optimum alignment it is not guaranteed.

In the default mode, A-GLAM repeats the annealing maximization procedure ten times from different starting points (ten runs). The rationale behind this is that if several of the runs converge to the same best alignment, the user has increased confidence that it is indeed the optimum alignment.

The individual score and its E-value in A-GLAM: The Gibbs sampling step produces an alignment whose overall score s is given by equation (1). Consider a window of length w that is about to be added to A-GLAM's alignment. Let $\delta_i(j)$ equal 1 if the window has nucleic acid j in position i, and 0 otherwise. The addition of the new window changes the overall score by

$$\Delta s = \sum_{i=1}^{w} \sum_{(j)} \delta_i(j) \left\{ \log_2 \left[\left(\frac{c_{ij} + a_j}{c + a} \right) / p_j \right] \right\} \qquad (2)$$

The score change corresponds to scoring the new window according to a position specific scoring matrix (PSSM) that assigns the "individual score"

$$s_i(j) = \log_2 \left[\left(\frac{c_{ij} + a_j}{c + a} \right) / p_j \right] \qquad (3)$$

to nucleic acid j in position i. Equation (3) represents a log-odds score for nucleic acid j in position i under an alternative hypothesis with probability $\left(c_{ij} + a_j\right)/(c + a)$ and a null hypothesis with

probability p_{ij}. PSI-BLAST *(40)* uses equation (3) to calculate E-values. The derivation through equation (2) confirms the PSSM in equation (3) as the natural choice for evaluating individual sequences.

The assignment of an E-value to a subsequence with a particular individual score is done as follows. Consider the alignment sequence containing the subsequence. Let n be the sequence length, and recall that w is the window size. If ΔS_i denotes the quantity in equation (2) if the final letter in the window falls at position i of the alignment sequence, then $\Delta S^* = \max\{\Delta S_i : i = w, \ldots, n\}$ is the maximum individual score over all sequence positions i. We assigned an E-value to the actual value $\Delta S^* = \Delta s^*$, as follows. Staden's method *(41)* yields $\mathbb{P}\{\Delta S_i \geq \Delta s^*\}$ (independent of i) under the null hypothesis of bases chosen independently and randomly from the frequency distribution $\{p_j\}$. The E-value $E = (n - w + 1)\mathbb{P}\{\Delta S_i \geq \Delta s^*\}$ is therefore the expected number of sequence positions with an individual score exceeding Δs^*. The factor $n - w + 1$ in E is essentially a multiple test correction.

More recently, the A-GLAM package has been improved to allow the identification of multiple instances of an element within a target sequence *(24)*. The optional "scanning step" after Gibbs sampling produces a PSSM given by equation (3). The new scanning step resembles an iterative PSI-BLAST search based on the PSSM (**Fig. 13.1**). First, it assigns an "individual score" to each subsequence of appropriate length within the input sequences using the initial PSSM. Second, it computes an E-value from each individual score to assess the agreement between the corresponding subsequence and the PSSM. Third, it permits subsequences with E-values falling below a threshold to contribute to the underlying PSSM, which is then updated using the Bayesian calculus. A-GLAM iterates its scanning step to convergence, at which point no new subsequences contribute to the PSSM. After convergence, A-GLAM reports predicted regulatory elements within each sequence in order of increasing E-values; users then have a statistical evaluation of the predicted elements in a convenient presentation. Thus, although the Gibbs sampling step in A-GLAM finds at most one regulatory element per input sequence, the scanning step can now rapidly locate further instances of the element in each sequence.

2.2. Hardware

The minimum hardware requirements are a personal computer with at least 512 MB of random access memory (RAM) connected to the Internet as well as access to a Linux or UNIX workstation where A-GLAM will be installed. The connectivity between the personal computer and the workstation is typically established by the Secure Shell (SSH) protocol, a widely used open source of the protocol available at http://www.openssh.org/.

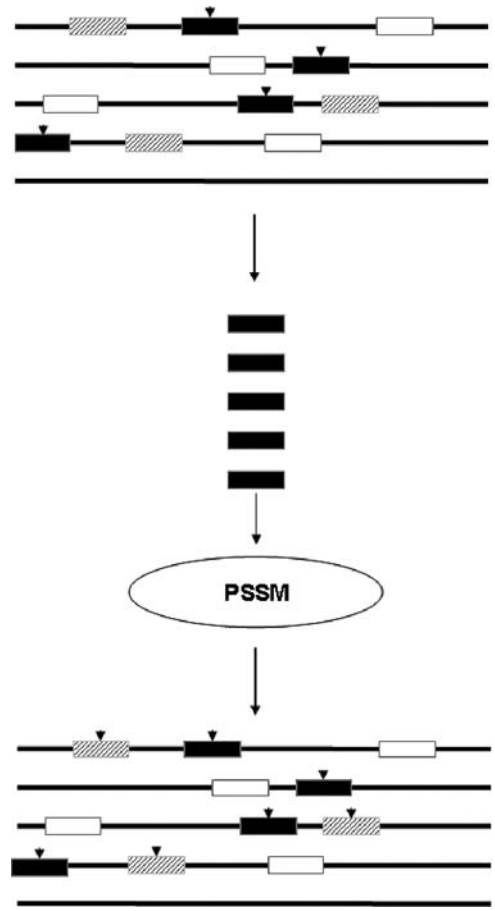


Fig. 13.1. **Strategy to find multiple motif instances in A-GLAM.** The Gibbs sampling identifies up to one motif per sequence (indicated by a *black box* and an *arrowhead*). The sequences are then used to construct a position specific score matrix (PSSM) that is used iteratively to discover multiple motif instances per sequence (indicated by *dashed boxes*).

2.3. Software

A modern version of the Perl programming language installed on the Linux or UNIX workstation freely available at http://www.perl.com/ will allow the user to parse A-GLAM's output. The A-GLAM package *(23)* freely available at http://ftp.ncbi.nih.gov/pub/spouge/papers/archive/AGLAM/ is currently available as source code and binary packages for the Linux operating system.

Installation of the Linux binary: get the executable from the FTP site and set execute permissions.

```
$chmod +x aglam
```

Installation from source: unpack the glam archive in a convenient location and compile A-GLAM.

```
$tar –zxvf aglam.tar.gz
$cd aglam
$make aglam
```

Then you could place the binary in your path: $HOME/bin or /usr/local/bin/.

2.4. Data Files

A-GLAM accepts input data in FASTA format containing the sequences to be analyzed. The FASTA format consists of one or more sequences identified by a line beginning with the ">" character that should include a unique identifier and a short description about the sequence. The next line(s) should contain the sequence string. A-GLAM expects the standard nucleic acid IUPAC code.

2.5. A-GLAM Options

Some important options to modify A-GLAM's behavior are described below:

```
$aglam <fasta_file.fa>
```

This command simply uses the standard Gibbs sampling procedure to find sequence motifs in "fasta_file.fa."

```
$aglam <fasta_file.fa> -n 30000 -a 8 -b 16 -j
```

These sets of commands instruct the program to search only the given strand of the sequences to find motifs of length between 8 and 16 bp. The flag *n* specifies the number of iterations performed in each of the ten runs. Low values of *n* are adequate when the problem size is small, i.e., when the sequences are short and more importantly there are few of them, but high values of *n* are needed for large problems. In addition, smaller values of *n* are sufficient when there is a strong alignment to be found, but larger values are necessary when there is no strong alignment, e.g., for finding the optimal alignment of random sequences. You will have to choose *n* on a case-by-case basis. This parameter also controls the tradeoff between speed and accuracy.

```
$aglam <fasta_file.fa> -i TATA
```

This important option sets the program to run in a "seed" oriented mode. The above command restricts the search to sequences that are TATA-like. Unlike the procedure followed in the standard Gibbs sampling algorithm, however, A-GLAM continues to align one exact copy of the "seed" in all "seed sequences." Therefore, A-GLAM uses the seed sequences to direct its search in the remaining non-seed "target sequences." Using this option leads to the global optimum quickly.

```
$aglam <fasta_file.fa> -l -k 0.05 -g 2000
```

Usable only with version 1.1. This set of commands instructs the program to find multiple motif instances in each input sequence via the scanning step (described above). Those instances that receive an E-value less than 0.05 are included in the PSSM. The search for multiple motifs is carried on until either (a) no new motifs are present or (b) the user-specified number of iterations (in this case, it is 2000) is attained, whichever comes first.

3. Example

The A-GLAM package includes documentation and test datasets. Here, we will use a dataset obtained from a large-scale chromatin immunoprecipitation in *Saccharomyces cerevisiae (2)*, combined with DNA microarrays *(42)* to detect interactions between transcription factors and a DNA sequence in vivo. The DNA sequence binding specificity of various transcription factors can then be inferred using A-GLAM on intergenic regions bound by a particular transcription factor. Here, we will use the intergenic regions bound by Snt2p (*see* **Note 1**).

3.1. Promoter Identification

The *Saccharomyces* Genome Database (SGD) maintains the most current annotations of the yeast genome (*see* http://www.yeastgenome.org/). The SGD FTP site contains the DNA sequences annotated as intergenic regions in FASTA format (available at ftp://genome-ftp.stanford.edu/pub/yeast/sequence/genomic_sequence/intergenic/), indicating the 5′ and 3′ flanking features. Additionally, a tab delimited file with the annotated features of the genome is necessary to determine the orientation of the intergenic regions relative to the genes (available at ftp://genome-ftp.stanford.edu/pub/yeast/chromosomal_feature/). The two files can be used to extract upstream intergenic regions. Additionally, there are a number of Web services that facilitate the identification of proximal promoter in mammalian genomes; these include TRED *(32)*, EPD *(33)*, and Promoser *(31)*.

3.2. Identification of cis-Regulatory Elements in Promoter Regions

Construct FASTA files for each of the promoters to be included in the analysis. The Perl programming language can be used in conjunction with BioPerl libraries (freely available at http://www.bioperl.org/) to generate files in FASTA format. In this particular example all relevant files can be found on the Fraenkel Web site at http://fraenkel.mit.edu//Harbison/release_v24.

The A-GLAM package has a number of options that can be used to adjust search parameters.

```
$aglam
Usage summary: aglam [ options] myseqs.fa
Options:
-h help: print documentation
-n end each run after this many iterations
without improvement (10000)
-r number of alignment runs (10)
-a minimum alignment width (3)
-b maximum alignment width (10000)
-j examine only one strand
-i word seed query ()
-f input file containing positions of the
motifs ()
-z turn off ZOOPS (force every sequence to
participate in the alignment)
-v print all alignments in full
-e turn off sorting individual sequences in an
alignment on p-value
-q pretend residue abundances = 1/4
-d frequency of width-adjusting moves (1)
-p pseudocount weight (1.5)
-u use uniform pseudocounts: each pseudocount =
p/4
-t initial temperature (0.9)
-c cooling factor (1)
-m use modified Lam schedule (default = geo-
metric schedule)
-s seed for random number generator (1)
-w print progress information after each
iteration
-l find multiple instances of motifs in each
sequence
-k add instances of motifs that satisfy the
cutoff e-value (0)
-g number of iterations to be carried out in
the post-processing step (1000)
```

Run A-GLAM to identify regulatory elements present in the promoter regions bound by Snt2p. A-GLAM uses sequences in FASTA format as input. There are 46 intergenic regions bound by Snt2p that were identified by ChIP-chip in a large-scale study *(2)*. These regions vary in length from 71 to 1,512 bp with an average of 398 bp. A-GLAM is able to identify statistically significant motifs for Snt2p and rank them according to their

individual *p*-values. A-GLAM has a number of useful command line options that can be adjusted to improve ab initio motif finding; in this example we have restricted the search to motifs no larger than 20 bp and instructed the program to find multiple instances of motifs in each sequence using a strategy that resembles an iterative PSI-BLAST search based on the PSSM constructed by the Gibbs sampling step *(24)*. The output of the A-GLAM program is presented in **Fig. 13.2**. In the default mode, A-GLAM repeats the annealing maximization procedure ten times from different starting points (ten runs). The rationale behind this is that if several of the runs converge to the same best alignment, the user has increased confidence that it is indeed the optimum alignment. The user can adjust the number of alignment runs by setting the –r flag (*see* **Note 2**). The number of iterations can also be adjusted for large datasets. The default value is set at 10,000 without alignment improvement, using the –n flag the number of iterations can be increased to extend coverage of the sequence space.

A-GLAM identifies candidate sequences that could serve as Snt2p binding sites. The candidate sequences found by A-GLAM are in agreement with previous findings where other motif finding algorithms were used *(2)* and **Fig. 13.3**. Additional examples where we have successfully used A-GLAM to complement experimental efforts for the identification of regulatory elements include motifs for Spt10p in yeast and the CREB-binding protein *(34, 35)*. In this particular example, the program constructs a PSSM using the sequences from the optimal alignment to find multiple instances (*see* **Note 3**). The multiple alignments produced by A-GLAM can be represented graphically by sequence logos *(43, 44)* (*see* **Note 4**).

4. Notes

1. The primary data can be obtained from the Fraenkel Laboratory Web site at http://fraenkel.mit.edu/Harbison/.
2. The number of alignment runs is 10 by default; however, the user can increase the number of runs to boost the confidence of the results. The user has the option –v to print all alignments generated in each run; by default A-GLAM will report only the highest scoring alignment.
3. Alternatively, the user could run A-GLAM without the –l flag and construct a position frequency matrix that in turn could be used to scan the target sequences for additional instances of the motif. The TFBS Perl modules for

```
$ aglam –b 20 -l SNT2_YPD.fsa

A-GLAM: Anchored Gapless Local Alignment of Multiple Sequences
Compiled on Feb 9 2008
aglam -l SNT2_YPD.fsa

Run 1...  25340 iterations
Run 2...  26770 iterations
Run 3...  22597 iterations
Run 4...  17786 iterations
Run 5...  23816 iterations
Run 6...  42556 iterations
Run 7...  19556 iterations
Run 8...  22526 iterations
Run 9...  23310 iterations
Run 10...  21531 iterations

! The sequence file was    [SNT2_YPD.fsa]
! Reading the file took    [0] secs
! Sequences in file    [46]
! Maximum possible alignment width    [142]
! Score    [400] bits
! Motif Width      [12]
! Runs    [10]

! Best possible alignment:

>iYNL182C 6.2046e-10              202  ATGGCGCTATCA   213  +    (10.24060) (1.714353e-02)
>iYBL075C 7.9181e-10              278  GCGGCGCTATCA   267  -    (12.62630) (3.136227e-04)
>iYIL160C 1.0190e-09              211  ACGGCGCTACCA   222  +    (14.16730) (2.230312e-05)
                                  208  AAGGCGCTATCA   197  -    (10.97000) (4.259169e-03)
>iYPR183W 1.6110e-09              237  GCGGCGCTACCA   248  +    (14.39810) (8.284745e-06)
>iYCR090C-1 2.2463e-09            575  ACGGCGCTATCA   564  -    (12.39550) (1.190739e-03)
>iYAL039C-0 5.6844e-09            281  GCGGCGCTACCA   292  +    (14.39810) (2.092311e-05)
>iYPR157W 1.0834e-08              343  GTGGCGCTATCA   332  -    (10.47150) (1.119393e-02)
>iYOL117W 1.2728e-08                      absent
>iYLR149C 1.4205e-08              252  ATGGCGCTACCA   263  +    (12.01250) (1.704126e-03)
>iYJL093C 3.7648e-08              279  GCGGCGCTATCA   268  -    (12.62630) (6.283269e-04)
>iYBR143C 2.6501e-07              202  ACGGCGCTATCA   213  +    (12.39550) (1.581019e-03)
>iYLR176C 1.6035e-06              221  GTGGCGCTACCA   232  +    (12.24330) (1.517043e-03)
>iYPR104C 6.0302e-06              420  ATGGCGCTATCA   431  +    (10.24060) (1.333119e-02)
>iYBR138C 9.2799e-06              203  GCGGCGCTAGCA   214  +    (12.42200) (4.809407e-04)
                                  206  CCGCCTCGGCCA   195  -    (8.225040) (4.975689e-02)
>tP(UGG)M 1.4586e-05              26   CCAGCTCGCCCC   15   -    (8.644490) (1.269803e-02)
                                  99   ACCACTAGACCA   110  +    (7.042530) (4.773258e-02)
>iYHR217C 1.7438e-05                      absent
>iYHR138C 3.9997e-05              145  TCGGCGCTACCA   134  -    (11.23880) (3.713288e-03)
>iYKL172W 4.5759e-05                      absent
>IntYGL103W 4.7991e-05                    absent
>tL(UAG)L2 4.9893e-05             21   ACCACTCGGCCA   10   -    (9.819350) (3.647425e-03)
>iYJR152W 5.1753e-05                      absent
>tS(AGA)M 6.7229e-05              35   CCTGCGCGGGCA   46   +    (9.031130) (6.321172e-03)
>tW(CCA)P 6.8308e-05              81   AAAGCTCTACCA   92   +    (10.76890) (1.315551e-03)
>snR128 9.5071e-05                        absent
>tI(UAU)L 9.5693e-05              26   GCAACGCGACCG   15   -    (8.373710) (1.822708e-02)
>iYCR090C-0 1.0515e-04                    absent
>IntYPL081W 1.1828e-04                    absent
>SNR190 1.1910e-04                162  CCGATTCGACCA   151  -    (7.925580) (4.569485e-02)
>tL(CAA)C 1.2473e-04              24   ACCGCTCGGCCA   13   -    (11.62350) (5.732354e-04)
>tP(AGG)C 1.3420e-04              24   CCGGCTCGCCCC   13   -    (9.501020) (4.510700e-03)
>tS(GCU)L 1.9681e-04                      absent
>tH(GUG)M 2.0224e-04                      absent
>SNR43 2.6811e-04                         absent
>tS(CGA)C 3.0856e-04              71   CCAGCGCGGGCA   60   -    (10.69280) (1.375755e-03)
>tK(UUU)P 3.1249e-04              55   AACGCTCTACCA   66   +    (10.63040) (1.330965e-03)
>tN(GUU)P 4.7144e-04              32   CCAACTTGGCCA   21   -    (7.907740) (2.015338e-02)
>tV(AAC)M3 4.8949e-04             60   CCGACTAGACCA   71   +    (7.828140) (1.674674e-02)
>tA(UGC)O 5.7662e-04              48   AGCGCGCTACCA   59   +    (9.775690) (2.504172e-03)
>tT(AGU)O2 6.8638e-04             60   CCAACTTGGCCA   71   +    (7.907740) (1.737360e-02)
>tR(UCU)M2 7.2321e-04             55   GACGCGTTGCCA   66   +    (9.845220) (2.902318e-03)
>iYLR228C-1 8.1088e-04                    absent
>tQ(UUG)L 8.1134e-04                      absent
>tC(GCA)P2 9.1920e-04             46   GCTGCGCTACCA   57   +    (11.88000) (2.284798e-04)
>iYDR261C-1 9.4038e-04                    absent
>SNR44 9.4060e-04                         absent
>tG(GCC)P2 9.5116e-04             26   CCAACGTTGCCA   37   +    (9.164880) (5.191352e-03)

! 34 sequences in alignment
! Residue abundances:Pseudocounts
! A = 0.311204:0.466806 C = 0.188796:0.283194 G = 0.188796:0.283194 T = 0.311204:0.466806
! Total Time to find best alignment    [13.92] secs
```

Fig. 13.2. **A-GLAM output for a set of sequences containing an SNT2p motif identified using ChIP-chip.** A-GLAM works by analyzing completely random alignment of the sequences and making small refinements over ten alignment runs with many iterations.

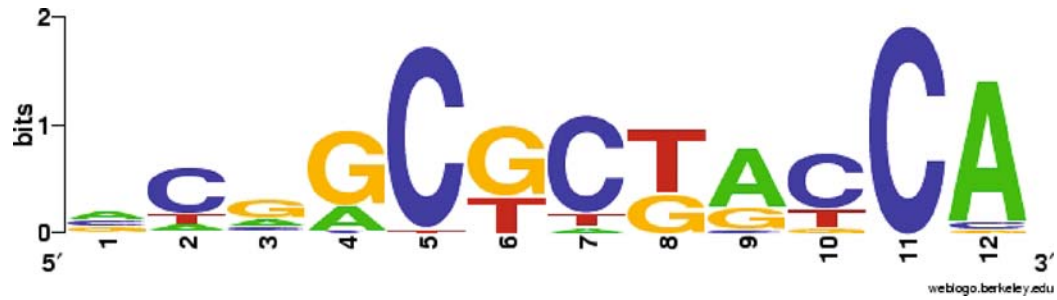


Fig. 13.3. **Snt2p regulatory motif identified with A-GLAM.** Sequence logo representation of the motif obtained from the ungapped multiple sequence alignment identified by A-GLAM.

transcription factor binding detection and analysis provide a flexible and powerful framework (available at http://tfbs.genereg.net/).

4. Other Web servers for logo generation include enoLOGOS (available on the Web at http://biodev.hgen.pitt.edu/enologos/) and Pictogram (http://genes.mit.edu/pictogram.html).

Acknowledgments

This research was supported by the Intramural Research Program of the NIH, NLM, NCBI.

References

1. Elnitski, L., Jin, V. X., Farnham, P. J., and Jones, S. J. (2006) Locating mammalian transcription factor binding sites: a survey of computational and experimental techniques. *Genome Res* **16,** 1455–64.
2. Harbison, C. T., Gordon, D. B., Lee, T. I., Rinaldi, N. J., Macisaac, K. D., Danford, T. W., Hannett, N. M., Tagne, J. B., Reynolds, D. B., Yoo, J., et al. (2004) Transcriptional regulatory code of a eukaryotic genome. *Nature* **431,** 99–104.
3. Bieda, M., Xu, X., Singer, M. A., Green, R., and Farnham, P. J. (2006) Unbiased location analysis of E2F1-binding sites suggests a widespread role for E2F1 in the human genome. *Genome Res* **16,** 595–605.
4. Cawley, S., Bekiranov, S., Ng, H. H., Kapranov, P., Sekinger, E. A., Kampa, D., Piccolboni, A., Sementchenko, V., Cheng, J., Williams, A. J., et al. (2004) Unbiased mapping of transcription factor binding sites along human chromosomes 21 and 22 points to widespread regulation of noncoding RNAs. *Cell* **116,** 499–509.
5. Guccione, E., Martinato, F., Finocchiaro, G., Luzi, L., Tizzoni, L., Dall' Olio, V., Zardo, G., Nervi, C., Bernard, L., and Amati, B. (2006) Myc-binding-site recognition in the human genome is determined by chromatin context. *Nat Cell Biol* **8,** 764–70.
6. Hughes, J. D., Estep, P. W., Tavazoie, S., and Church, G. M. (2000) Computational identification of *cis*-regulatory elements associated with groups of functionally related genes in *Saccharomyces cerevisiae*. *J Mol Biol* **296,** 1205–14.

7. Workman, C. T., and Stormo, G. D. (2000) ANN-Spec: a method for discovering transcription factor binding sites with improved specificity. *Pac Symp Biocomput* **5**, 467–78.
8. Hertz, G. Z., and Stormo, G. D. (1999) Identifying DNA and protein patterns with statistically significant alignments of multiple sequences. *Bioinformatics* **15,** 563–77.
9. Frith, M. C., Fu, Y., Yu, L., Chen, J. F., Hansen, U., and Weng, Z. (2004) Detection of functional DNA motifs via statistical over-representation. *Nucleic Acids Res* **32,** 1372–81.
10. Ao, W., Gaudet, J., Kent, W. J., Muttumu, S., and Mango, S. E. (2004) Environmentally induced foregut remodeling by PHA-4/FoxA and DAF-12/NHR. *Science* **305,** 1743–6.
11. Bailey, T. L., and Elkan, C. (1994) Fitting a mixture model by expectation maximization to discover motifs in biopolymers. *Proc Int Conf Intell Syst Mol Biol* **2,** 28–36.
12. Eskin, E., and Pevzner, P. A. (2002) Finding composite regulatory patterns in DNA sequences. *Bioinformatics* **18 Suppl 1,** S354–63.
13. Thijs, G., Lescot, M., Marchal, K., Rombauts, S., De Moor, B., Rouze, P., and Moreau, Y. (2001) A higher-order background model improves the detection of promoter regulatory elements by Gibbs sampling. *Bioinformatics* **17,** 1113–22.
14. Régnier, M., and Denise, A. (2004) Rare events and conditional events on random strings. *Discrete Math Theor Comput Sci* **6,** 191–214.
15. Favorov, A. V., Gelfand, M. S., Gerasimova, A. V., Ravcheev, D. A., Mironov, A. A., and Makeev, V. J. (2005) A Gibbs sampler for identification of symmetrically structured, spaced DNA motifs with improved estimation of the signal length. *Bioinformatics* **21,** 2240–5.
16. Pavesi, G., Mereghetti, P., Zambelli, F., Stefani, M., Mauri, G., and Pesole, G. (2006) MoD Tools: regulatory motif discovery in nucleotide sequences from co-regulated or homologous genes. *Nucleic Acids Res* **34,** W566–70.
17. Pavesi, G., Zambelli, F., and Pesole, G. (2007) WeederH: an algorithm for finding conserved regulatory motifs and regions in homologous sequences. *BMC Bioinformatics* **8,** 46.
18. Sinha, S., and Tompa, M. (2003) YMF: A program for discovery of novel transcription factor binding sites by statistical overrepresentation. *Nucleic Acids Res* **31,** 3586–8.
19. Blanchette, M., Bataille, A. R., Chen, X., Poitras, C., Laganiere, J., Lefebvre, C., Deblois, G., Giguere, V., Ferretti, V., Bergeron, D., et al. (2006) Genome-wide computational prediction of transcriptional regulatory modules reveals new insights into human gene expression. *Genome Res.* **16,** 656–68.
20. Tompa, M., Li, N., Bailey, T. L., Church, G. M., De Moor, B., Eskin, E., Favorov, A. V., Frith, M. C., Fu, Y., Kent, W. J., et al. (2005) Assessing computational tools for the discovery of transcription factor binding sites. *Nat Biotechnol* **23,** 137–44.
21. Ohler, U., and Niemann, H. (2001) Identification and analysis of eukaryotic promoters: recent computational approaches. *Trends Genet* **17,** 56–60.
22. Marino-Ramirez, L., Spouge, J. L., Kanga, G. C., and Landsman, D. (2004) Statistical analysis of over-represented words in human promoter sequences. *Nucleic Acids Res* **32,** 949–58.
23. Tharakaraman, K., Marino-Ramirez, L., Sheetlin, S., Landsman, D., and Spouge, J. L. (2005) Alignments anchored on genomic landmarks can aid in the identification of regulatory elements. *Bioinformatics* **21 Suppl 1,** i440–8.
24. Tharakaraman, K., Marino-Ramirez, L., Sheetlin, S., Landsman, D., and Spouge, J. L. (2006) Scanning sequences after Gibbs sampling to find multiple occurrences of functional elements. *BMC Bioinformatics* **7,** 408.
25. Goffeau, A., Barrell, B. G., Bussey, H., Davis, R. W., Dujon, B., Feldmann, H., Galibert, F., Hoheisel, J. D., Jacq, C., Johnston, M., et al. (1996) Life with 6000 genes. *Science* **274,** 546, 563–47.
26. Lander, E. S., Linton, L. M., Birren, B., Nusbaum, C., Zody, M. C., Baldwin, J., Devon, K., Dewar, K., Doyle, M., FitzHugh, W., et al. (2001) Initial sequencing and analysis of the human genome. *Nature* **409,** 860–921.
27. Levine, M., and Tjian, R. (2003) Transcription regulation and animal diversity. *Nature* **424,** 147–51.
28. Carninci, P., Waki, K., Shiraki, T., Konno, H., Shibata, K., Itoh, M., Aizawa, K., Arakawa, T., Ishii, Y., Sasaki, D., et al. (2003) Targeting a complex transcriptome: the construction of the mouse full-length cDNA encyclopedia. *Genome Res* **13,** 1273–89.
29. Kimura, K., Wakamatsu, A., Suzuki, Y., Ota, T., Nishikawa, T., Yamashita, R.,

Yamamoto, J., Sekine, M., Tsuritani, K., Wakaguri, H., et al. (2006) Diversification of transcriptional modulation: large-scale identification and characterization of putative alternative promoters of human genes. *Genome Res* **16,** 55–65.

30. Suzuki, Y., Yamashita, R., Sugano, S., and Nakai, K. (2004) DBTSS, DataBase of Transcriptional Start Sites: progress report 2004. *Nucleic Acids Res* **32,** D78–81.
31. Halees, A. S., and Weng, Z. (2004) PromoSer: improvements to the algorithm, visualization and accessibility. *Nucleic Acids Res* **32,** W191–4.
32. Jiang, C., Xuan, Z., Zhao, F., and Zhang, M. Q. (2007) TRED: a transcriptional regulatory element database, new entries and other development. *Nucleic Acids Res* **35,** D137–40.
33. Schmid, C. D., Perier, R., Praz, V., and Bucher, P. (2006) EPD in its twentieth year: towards complete promoter coverage of selected model organisms. *Nucleic Acids Res* **34,** D82–5.
34. Eriksson, P. R., Mendiratta, G., McLaughlin, N. B., Wolfsberg, T. G., Marino-Ramirez, L., Pompa, T. A., Jainerin, M., Landsman, D., Shen, C. H., and Clark, D. J. (2005) Global regulation by the yeast Spt10 protein is mediated through chromatin structure and the histone upstream activating sequence elements. *Mol Cell Biol* **25,** 9127–37.
35. Riz, I., Akimov, S. S., Eaker, S. S., Baxter, K. K., Lee, H. J., Marino-Ramirez, L., Landsman, D., Hawley, T. S., and Hawley, R. G. (2007) TLX1/HOX11-induced hematopoietic differentiation blockade. *Oncogene* **26,** 4115–23.
36. van Helden, J., andre, B., and Collado-Vides, J. (1998) Extracting regulatory sites from the upstream region of yeast genes by computational analysis of oligonucleotide frequencies. *J Mol Biol* **281,** 827–42.
37. Lawrence, C. E., Altschul, S. F., Boguski, M. S., Liu, J. S., Neuwald, A. F., and Wootton, J. C. (1993) Detecting subtle sequence signals: a Gibbs sampling strategy for multiple alignment. *Science* **262,** 208–14.
38. Wasserman, W. W., and Sandelin, A. (2004) Applied bioinformatics for the identification of regulatory elements. *Nat Rev Genet* **5,** 276–87.
39. Marino-Ramirez, L., Jordan, I. K., and Landsman, D. (2006) Multiple independent evolutionary solutions to core histone gene regulation. *Genome Biol* **7,** R122.
40. Altschul, S. F., Madden, T. L., Schaffer, A. A., Zhang, J., Zhang, Z., Miller, W., and Lipman, D. J. (1997) Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Res* **25,** 3389–402.
41. Staden, R. (1989) Methods for calculating the probabilities of finding patterns in sequences. *Comput Appl Biosci* **5,** 89–96.
42. Ren, B., Robert, F., Wyrick, J. J., Aparicio, O., Jennings, E. G., Simon, I., Zeitlinger, J., Schreiber, J., Hannett, N., Kanin, E., et al. (2000) Genome-wide location and function of DNA binding proteins. *Science* **290,** 2306–9.
43. Schneider, T. D., and Stephens, R. M. (1990) Sequence logos: a new way to display consensus sequences. *Nucleic Acids Res* **18,** 6097–100.
44. Crooks, G. E., Hon, G., Chandonia, J. M., and Brenner, S. E. (2004) WebLogo: a sequence logo generator. *Genome Res* **14,** 1188–90.

Chapter 14

Analysis of Genomic DNA with the UCSC Genome Browser

Jonathan Pevsner

Abstract

Genomic DNA is being sequenced and annotated at a rapid rate, with terabases of DNA currently deposited in GenBank and other repositories. Genome browsers provide an essential collection of resources to visualize and analyze chromosomal DNA. The University of California, Santa Cruz (UCSC) Genome Browser provides annotations from the level of single nucleotides to whole chromosomes for four dozen metazoan and other species. The Genome Browser may be used to address a wide range of problems in bioinformatics (e.g., sequence analysis), comparative genomics, and evolution.

Key words: Chromosome, DNA sequence, gene, annotation, genomic variation.

1. Introduction

Genomic DNA is organized into chromosomes. Each eukaryotic species has a characteristic number of chromosomes packaged into the nucleus; in humans there are 23 pairs. As genomes continue to be sequenced, it becomes increasingly important to organize different categories of information about chromosomes including their raw sequence, their annotated sequence (e.g., the number and location of genes and repetitive DNA elements), their variation (e.g., across subpopulations of a species), and their dynamic regulation (e.g., the occurrence of chromosomal deletions, duplications, or other structural changes). Genome browsers centralize information about chromosomes at scales from single nucleotides to hundreds of millions of bases.

David Posada (ed.), *Bioinformatics for DNA Sequence Analysis,* Methods in Molecular Biology 537

DOI 10.1007/978-1-59745-251-9_14

There are currently three principal genome browsers that focus on eukaryotic genomes *(1)*. Each offers access to completed (reference) genomes as well as working assemblies that are not yet finished. First, the UCSC genome browser *(2)*, which is the focus of this chapter. Second, the National Center for Biotechnology (NCBI) offers a Map Viewer for a broad range of vertebrates, invertebrates, fungi, plants, and protozoa *(3)*. Third, the Ensembl Project *(4)* annotates and displays genomic data from a broad range of species, with an emphasis on vertebrates *(5)*. Ensembl is a collaboration between the European Molecular Biology Laboratory/European Bioinformatics Institute and the Wellcome Trust Sanger Institute.

All three of these genome browsers provide complementary views of the genome. They share a common underlying source of raw DNA sequence data and a common set of genome assemblies. They differ in how they annotate each genome (e.g., they map different databases to each genome), visualize information, and provide genome analysis tools. The UCSC Genome Browser has a key role in facilitating comparative and other genomic analyses. In particular, the many dozens of annotation tracks provided to the site by the external research community allow a genome-wide view of many biological phenomena. The bioinformatics resources of the site further stimulate exploration and discovery of features in vertebrate and other genomes.

2. Program Usage

The UCSC Genome Bioinformatics website is http://genome.ucsc.edu/. This website is accessible with standard browsers such as Internet Explorer and Mozilla Firefox. From the main site, users can access the Genome Browser *(2, 6)* and the Table Browser *(7)* as well as other resources.

The Genome Browser is the principal site for visualizing information, with a window size of a single chromosome or subset of a chromosome. A key first step is to specify a genome assembly to view, such as human or mouse. For human, several different builds are available (e.g., May 2004 and March 2006); each build contains different annotation. The main display window of the Genome Browser shows annotation tracks, which contain genomic data (such as the position of an exon or repetitive DNA element) that are mapped to specified chromosomal loci (that is, genomic positions). Examples 3.1–3.8 below all use the Genome Browser.

The Table Browser is complementary to the Genome Browser. The two tools share the same underlying database (from a particular genome assembly) and so provide consistent information. The Table Browser allows fields of

information to be viewed and exported in a tabular form. In one standard application, a region that is viewed in the Genome Browser may have features of interest (such as known genes as well as some repetitive element), and by following the link to the Table Browser one may readily obtain a list of those features including their genomic positions. Additionally, features may be cross-indexed, for example, allowing the user to select known genes that are within a certain distance of a particular repetitive element. Many of the tables underlying the Table Browser are positional in that they have specified start and stop coordinates. Other tables are non-positional (e.g., tables relating gene names to accession numbers). Examples 3.3, 3.4, and 3.6 highlight the use of the Table Browser.

Another powerful feature of the UCSC Genome and Table browser is its ability to incorporate data supplied by the user. For example, one may have performed a microarray experiment and generated some values for the expression levels of 20,000 transcripts or the properties of one million single nucleotide polymorphisms (SNPs). Such data can be placed in a spreadsheet, formatted as described below (*see* **Sections 3.3** and **3.8**), and uploaded to the UCSC website for display and analysis in the Genome Browser and/or Table Browser. A text editor can be helpful for creating custom annotation tracks). An example is Crimson Editor (a source code editor for the Windows platform available as freeware at http://www.crimsoneditor.com/). Such an editor can process data sets having millions of rows. Alternatively, a spreadsheet program such as Microsoft® Excel or an editor such as Microsoft® Notepad can be used to create custom tracks.

3. Examples

3.1. Characterizing a Human Gene: Text-Base Searching

Many searches begin with the name of a known gene. We will use the *HOXA1* gene as an example. The *HOX* genes encode transcription factors that regulate the anterior–posterior patterning of tissues along the body axis of vertebrates during development *(8)*. For many vertebrate species, the *HOX* genes are organized into four families (A, B, C, D) in clusters. Within a cluster, these genes are organized in a fashion that is collinear with their spatial and temporal expression pattern along the anterior–posterior axis of an embryo. In humans, the four *HOX* gene clusters occur on chromosomes 7, 17, 12, and 2, respectively.

From the home page of the UCSC Genome Bioinformatics site, select Genome Browser. Set the clade to vertebrate, the genome to

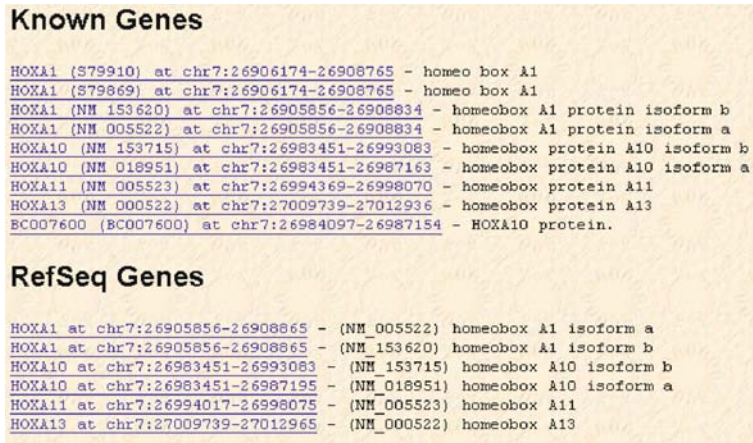


Fig. 14.1. A portion of the result for a text-based search (hoxa1) is shown. The appropriate link can be selected by name (e.g., HOXA1), genomic position, or accession number. RefSeq or non-RefSeq accession numbers are provided. The RefSeq project provides high quality curation and its accession numbers for DNA sequences are distinguished by having the identifier "NM_" preceding a string of numbers.

Human, and the assembly to May 2004 (*see* **Note 1**). Under "position or search term" type hoxa1. The output (**Fig. 14.1**) includes a set of known genes and reference sequence (RefSeq) genes (*see* **Note 2**). The UCSC "Known Genes" *(9)* are selected based on a fully annotated process that relies on the UniProt database *(10)* and GenBank mRNA data *(11)*. Some of the entries (e.g., S79910 for HOXA1) have accession numbers that are not part of the RefSeq project. A separate section shows the RefSeq genes; in this case, the sole entries for HOXA1 are NM_005522 (for isoform a) and NM_153620 (for isoform b). It may be helpful to obtain further information about a gene from external sources. For example, the NCBI Entrez Gene project (available at http://www.ncbi.nlm.nih.gov) includes a summary of the *HOX1A* gene noting that only the a isoform contains the homeodomain region implicated in transcriptional regulation.

Click the link for RefSeq isoform a (HOXA1 at chr7:26905856–26908865). This links to the UCSC Genome Browser entry for the *HOXA1* gene on chromosome 7 (**Fig. 14.2**). This is the main visualization tool of the UCSC website. The view can be zoomed in or out (*see* arrow A). The position is listed, giving the coordinates on chromosome 7; if different coordinates are entered, one can jump to a different genomic view. The size of the displayed window (i.e., the *x*-axis) is given as 3,010 base pairs (arrow B). An ideogram is displayed (arrow C), showing the chromosome band on the short arm of chromosome 7 to which this gene is assigned. The main display window (arrow D) contains a variety of annotation tracks, arranged horizontally. The many options for adding and displaying tracks are what provide the website with its depth and flexibility.

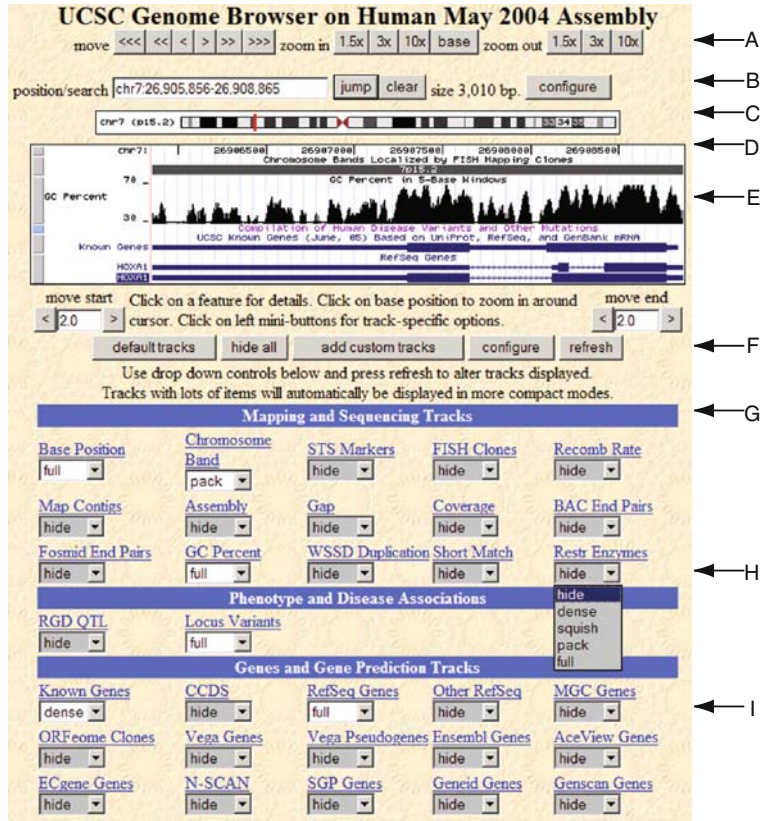


Fig. 14.2. Genome Browser view for HOXA1, showing the main features. The *top* portion includes zoom controls (*arrow* **A**), a box to select the chromosome and position (**B**), and an ideogram (**C**) corresponding to a chromosome and its bands. The main graphical viewer includes rows with various tracks that can be added, removed, and displayed in multiple formats. The *top* row (**D**) includes the base pair position; by clicking, one zooms in threefold. The graphical display offers a variety of viewing formats, e.g., the guanine + cytosine (GC) content in five-base windows (**E**). Custom tracks can be loaded and the display can be refreshed (**F**). The *bottom* portion of the Genome Browser includes many dozens of annotation tracks, organized into basic categories (**G**) such as Mapping and Sequencing Tracks. Clicking on a particular entry (such as GC Percent, *arrow* **H**) links to detailed information about that track; a variety of display options are available (hide, dense, squish, pack, full). For entries that do not have information in a particular window, such as Locus Variants in this case, the main display window shows the header (Compilation of Human Disease Variants) without further information. Known Genes and RefSeq genes are displayed (*arrow* **I**) at the dense and full settings.

Along the top row (arrow D), the physical map position is given (by clicking in that region, one can zoom in threefold). Below it the chromosome band is represented, followed by the GC percent (guanine and cytosine percent) in 5-base windows. One can add custom tracks (arrow F; discussed below). The main categories of annotation tracks vary according to the particular genome build, but for the human May 2004 assembly they include the following: (1) mapping

and sequencing tracks (arrow G); (2) phenotype and disease assocations; (3) genes and gene prediction; (4) messenger RNA (mRNA) and expressed sequence tags (ESTs); (5) expression and regulation; (6) comparative genomics; and (7) variation and repeats. Six further categories are related to the ENCODE project (discussed in **Section 3.6** below) in which 1% of the human genome has been characterized in extreme depth.

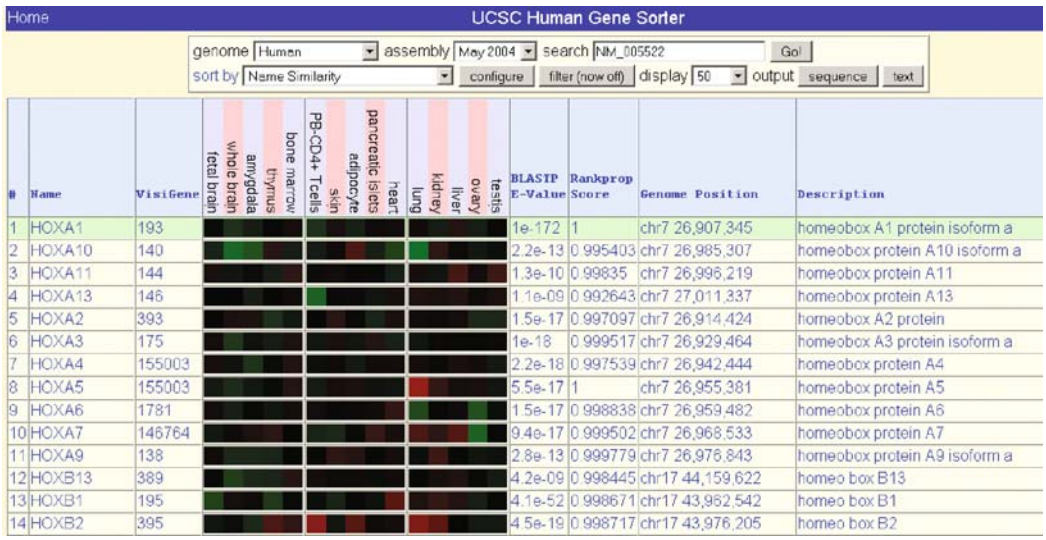
The annotation tracks thus involve a broad range of information. These tracks can be hidden or displayed with varying compactness of information, from dense to full (**Fig. 14.2**, arrow H). Note that the GC Percent track has been set to "full" in this example. By clicking on the label "GC Percent" one accesses a page with detailed information on that track, including a variety of display options, descriptions, information about the source of the information (whether generated by the UCSC developers or contributed by members of the external research community), and relevant literature citations. For tracks that do not contain information in this particular region, such as disease-associated locus variants, the title is displayed without any information below it (**Fig. 14.2**, below arrow E).

3.2. Characterizing a Human Gene: Text-Base Searching with the Gene Sorter

In studying a particular gene or protein, a common problem is to determine how many genes form parts of a family. How many human *HOX* genes are there, and to what chromosomes are they assigned?

From the home page (http://genome.ucsc.edu), select Gene Sorter (from the top bar or left sidebar). Set the genome to human, and the assembly to May 2004. Enter hox and click go. Select HOXA1 from the list of results. You can sort the output over a dozen different ways using a pull-down menu. Select "Protein Homology – BLASTP" and the top 20 table entries are *HOX* family members that (by inspection) are assigned to chromosomes 7 (*HOXA* family), 17 (*HOXB*), 12 (*HOXC*), and 2 (*HOXD*). Next sort by "Name Similarity" and the top 37 entries are all from these same four *HOX* gene clusters (**Fig. 14.3**). This strategy provided a more comprehensive list of *HOX* genes, but (in contrast to sorting by BLASTP) the results are not sorted by similarity to *HOXA1*. This provides an example of the flexibility offered by different sorting and output options.

Inspect the columns in the table (**Fig. 14.3**). Click "configure" to see the available categories; currently, there are 50 options. We will examine several of them. VisiGene entries (**Fig. 14.3**, third column) provide links to in situ hybridization images, showing the localization of the RNA transcript for a gene across particular regions of the mouse and at specific developmental stages. The data are derived from the Mouse Genome Informatics website (http://www.informatics.jax.org). A variety of related gene



#	Name	VisiGene	fetal brain	whole brain	amygdala	thymus	bone marrow	PB-CD4+ Tcells	skin	adipocyte	pancreatic islets	heart	lung	kidney	liver	ovary	testis	BLASTP E-Value	Rankprop Score	Genome Position	Description
1	HOXA1	193																1e-172	1	chr7 26,907,345	homeobox A1 protein isoform a
2	HOXA10	140																2.2e-13	0.995403	chr7 26,985,307	homeobox protein A10 isoform a
3	HOXA11	144																1.3e-10	0.99835	chr7 26,996,219	homeobox protein A11
4	HOXA13	146																1.1e-09	0.992643	chr7 27,011,337	homeobox protein A13
5	HOXA2	393																1.5e-17	0.997097	chr7 26,914,424	homeobox A2 protein
6	HOXA3	175																1e-18	0.999517	chr7 26,929,464	homeobox A3 protein isoform a
7	HOXA4	155003																2.2e-18	0.997539	chr7 26,942,444	homeobox protein A4
8	HOXA5	155003																5.5e-17	1	chr7 26,955,381	homeobox protein A5
9	HOXA6	1781																1.5e-17	0.998838	chr7 26,959,482	homeobox protein A6
10	HOXA7	146764																9.4e-17	0.999502	chr7 26,968,533	homeobox protein A7
11	HOXA9	138																2.8e-13	0.999779	chr7 26,976,843	homeobox protein A9 isoform a
12	HOXB13	389																4.2e-09	0.998445	chr17 44,159,622	homeo box B13
13	HOXB1	195																4.1e-52	0.998671	chr17 43,962,542	homeo box B1
14	HOXB2	395																4.5e-19	0.998717	chr17 43,976,205	homeo box B2

Fig. 14.3. The UCSC Human Gene Sorter summarizes information about genes and their expression patterns.

expression data are provided based on microarray data (**Fig. 14.3**, columns to the right of VisiGene). These columns are color-coded, with default settings of red corresponding to relatively high RNA transcript levels, and green corresponding to low transcript levels across of variety of tissues.

The "Genome Position" column indicates the chromosomal locus of each gene. In this case, the full table clearly shows the genomic loci of the *HOXA*, *HOXB*, *HOXC*, and *HOXD* genes. As an example of another application of the Gene Sorter, one can display Protein Data Bank entries to show which genes on the list encode proteins that have their three-dimensional structures solved by X-ray crystallography or nuclear magnetic resonance spectroscopy.

3.3. Querying the Database with Molecular Sequences: BLAT

The Basic Local Alignment Search Tool (BLAST) at NCBI is a fundamental program used to search a protein or DNA query against a protein or DNA database *(12, 13)*. The UCSC Genome Bioinformatics site offers a BLAST-like Alignment Tool (BLAT) that is specialized for the extremely rapid search of a protein or DNA query against a genomic DNA database *(14)*. BLAT was constructed in particular to align human expressed sequence tags or mRNA to genomic DNA, but it has many applications.

BLAST searches are performed using local (rather than global) alignments of a query to a database. Rather than searching the database exhaustively, a query is parsed into words (at NCBI the default size is 11 nucleotides for DNA or 3 amino acids for protein

queries). For DNA queries, these are exactly matched to the database; for protein queries, matches above a threshold are selected. Matches to database sequences are then extended using dynamic programming to identify high-scoring local alignments. BLAT takes an opposite approach, building an index of an entire genomic DNA database and then using it to scan a query. This indexing augments the speed of a BLAT search, even with very large DNA queries (*see* **Note 3**). BLAT also efficiently combines neighboring alignments, often reporting a unified result rather than separate, adjacent database matches. We will illustrate the use of BLAT with HOXA1 protein, although one can also choose to use a DNA query. This example will also refer to the UCSC Proteome Browser and the Table Browser.

Use the BLAT to search for *HOXA1* homologs. First, select the HOXA1 isoform a protein sequence (335 residues) in the fasta format (*see* **Note 4**). This can be obtained from a source such as Entrez Protein at NCBI, allowing the selection of a RefSeq entry.

```
>gi|5031761|ref|NP_005513.1| homeobox A1 isoform
a [ Homo sapiens]
MDNARMNSFLEYPILSSGDSGTCSARAYPSDHRITTFQSCAVSANSCG
GDDRFLVGRGVQIGSPHHHHHHHHHHPQPATYQTSGNLGVSYSHSSCG
PSYGSQNFSAPYSPYALNQEADVSGGYPQCAPAVYSGNLSSPMVQHHH
HHQGYAGGAVGSPQYIHHSYGQEHQSLALATYNNSLSPLHASHQEACR
SPASETSSPAQTFDWMKVKRNPPKTGKVGEYGYLGQPNAVRTNFTTKQ
LTELEKEFHFNKYLTRARRVEIAASLQLNETQVKIWFQNRRMKQKKRE
KEGLLPISPATPPGNDEKAEESSEKSSSSPCVPSPGSSTSDTLTTSH
```

One can also obtain this sequence from the UCSC Proteome Browser. From the home page of the UCSC Genome Bioinformatics site, select Proteome Browser and enter the query hoxa1. Several human entries are displayed, including the full-length protein (the accession is P49639) and several shorter isoforms. RefSeq identifiers are not currently available. For the protein entry of interest, the Proteome Browser displays a summary of some physical properties of the protein (e.g., its molecular weight, isoelectric point, and amino acid composition), a link to the results of a BLAT search, links to other protein databases such as the protein family database Pfam and the Structural Classification of Proteins database SCOP, and the sequence of HOXA1 in the fasta format. Select and copy the amino acid sequence.

Next, visit BLAT (accessible from the home page, http://genome.ucsc.edu). Set the genome to human, the assembly to May 2004, and leave the other options at their default settings

(including the output type "hyperlink"). Paste in the amino acid query sequence. Inspect the output, which consists of a table with hyperlinked entries (**Fig. 14.4a**). Note that the best score is to a chromosome 7 match (corresponding to the *HOXA1* gene). There are additional matches on chromosomes such as 2, 12, 17, as well as other regions of chromosome 7.

(a)

BLAT Search Results

ACTIONS	QUERY	SCORE	START	END	QSIZE	IDENTITY	CHRO	STRAND	START	END	SPAN
browser details	YourSeq	998	1	335	335	99.8%	7	+-	26907302	26908771	1470
browser details	YourSeq	184	202	290	335	87.4%	17	+-	43962022	43962737	716
browser details	YourSeq	141	231	293	335	87.4%	2	++	176880081	176880269	189
browser details	YourSeq	81	233	285	335	75.5%	7	+-	26914119	26914277	159
browser details	YourSeq	81	233	285	335	75.5%	17	+-	43975903	43976061	159
browser details	YourSeq	72	243	286	335	77.3%	7	+-	26967900	26968031	132
browser details	YourSeq	72	243	286	335	77.3%	7	+-	26958583	26958714	132
browser details	YourSeq	72	233	286	335	72.3%	7	+-	26942231	26942392	162
browser details	YourSeq	72	243	286	335	77.3%	12	++	52709768	52709899	132
browser details	YourSeq	72	233	286	335	72.3%	12	++	52714349	52714510	162
browser details	YourSeq	69	232	286	335	71.0%	2	++	176861799	176861963	165

(b)

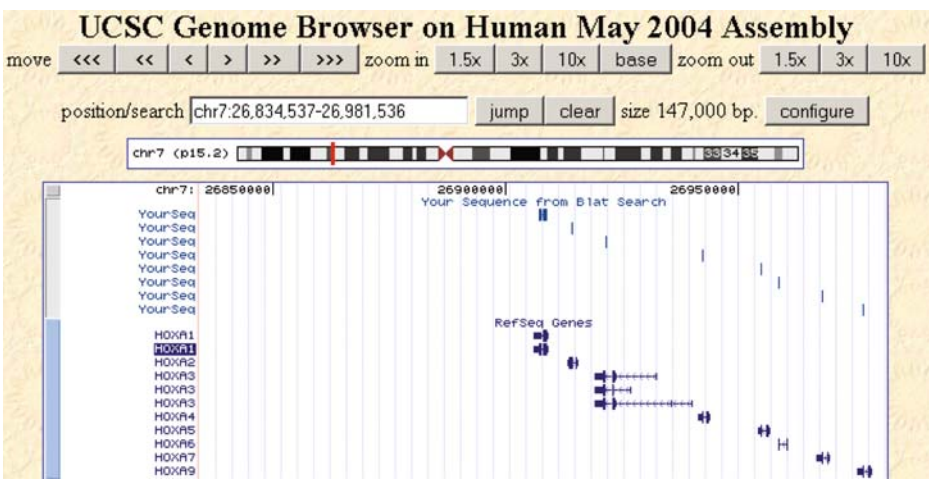


Fig. 14.4. Results of a BLAT search using human HOXA1 protein as a query. (**a**) The output format is set to hyperlink, providing links to both the UCSC Genome Browser and to a detailed pairwise alignment between the query and each database match (arranged in rows). The columns include the alignment score, start and end of the aligned query region, the query size (335 amino acids), the percent amino acid identity to each match, and information about each genomic match: the chromosome, strand, genomic start and end coordinates, and the span. By inspection of this table, only the first genomic match aligns fully with the query, while the other genomic sequences align with the query in subregions beginning at amino acids 202–243 and ending at residues 285–293. This corresponds to a conserved domain in HOX proteins. (**b**) Genome Browser view of BLAT search results on human chromosome 7, showing matches of the query (HOXA1) to a series of eight *HOXA* genes. For this search, genomic DNA was translated in six frames and all predicted amino acid sequences were compared to the query. The tracks "Blat Sequence" and "RefSeq Genes" are set to "full." Note that the HOXA1 BLAT query matches all of the protein-coding portion of the *HOXA1* gene, and portions of the other *HOX* genes that encode a domain shared with HOXA1.

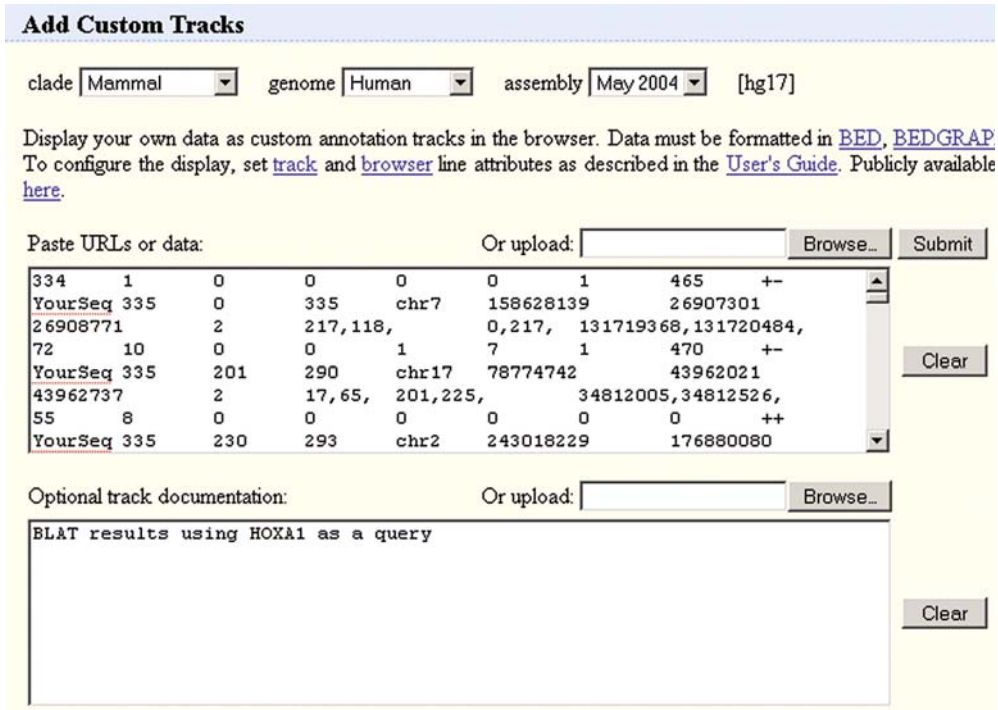
For the top entry having the best score, click on "browser." A track appears with the header "Your Sequence from BLAT Search," and a window size of 1,470 base pairs. Note that the BLAT query was a protein, so the matches in the browser are to the protein-coding exons (rather than to the introns or untranslated regions).

View additional genes that are paralogous to *HOXA1*. Press the "zoom out 10x" button twice to zoom out to a window size of 147,000 base pairs (chr7:26,834,537–26,981,536). The BLAT result and the RefSeq genes are displayed in the "full" mode (**Fig. 14.4b**). Your BLAT query appears in places where it matches other DNA sequences that encode significantly related proteins. By visual inspection, the query sequence from the BLAT search (i.e., *HOXA1*) significantly matched a group of other HOX genes, i.e., *HOXA2, HOXA3, HOXA4, HOXA5, HOXA6, HOXA7*, and *HOXA9*. Note that the RefSeq genes include multiple entries (two lines are assigned to *HOXA1* and three lines are assigned to *HOXA3*) corresponding to distinct RNA transcripts.

While visual inspection of the results can be useful, it is sometimes necessary to determine the exact nature of the results displayed in the genome browser. The UCSC Table Browser is thus an essential complement to the Genome Browser. We will next determine exactly which chromosome 7 genes encode proteins that match the BLAT query. To do this, we will create a table containing the BLAT results. Select the HOXA1 protein sequence in the fasta format. Return to the BLAT search page (for human, May 2004) and enter the query sequence. For output type, select "psl no header." (PSL is an abbreviation for the pat space layout format; other formats are discussed in **Section 3.8** below.) The output consists of a table in which the rows correspond to matches to the query sequence and the columns contain assorted information about the genomic location of those matches. Use the edit and copy functions to select the output. Along the top bar, click "Genomes" and make sure that the settings are to the May 2004 assembly of the human genome. Click "add custom tracks" and paste in your psl to the box labeled "Paste URLs or data." In the box below, labeled "Optional track documentation," enter the phrase "BLAT results using HOXA1 as a query." The web page is shown in **Fig. 14.5a**. Click submit, leading to a page titled Manage Custom Tracks (**Fig. 14.5b**). Here, click "go to table browser." The result is shown in **Fig. 14.6**.

Now that we have created a table, we will use the Table Browser to gather information about the BLAT search results. Note that the region is set to "genome" (**Fig. 14.6**, arrow B). By clicking "summary/statistics" (arrow D), you can see that there are 36 entries for the entire genome. To list the chromosome 7

(a)



(b)

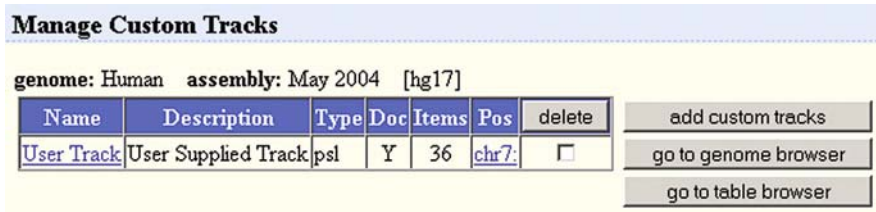


Fig. 14.5. The creation of custom tracks from BLAT output. (**a**) The "Add Custom Tracks" option allows you to upload or paste data such as the output of a BLAT search. Optional track documentation is displayed when this track is selected within the Genome Browser. (**b**) The Manage Custom Tracks option is an entry to the Genome Browser or the Table Browser. Multiple custom tracks can be loaded.

entries, set the region to "position" (using the radio button) and chr7 (arrow B). Click "lookup" and the position is now set to chr7:1–158628139 (i.e., the entire chromosome 7). Now, by clicking "get output" the ten chromosome 7 genes that match the HOXA1 query are listed. What are the names of these genes? Click intersection (arrow C). Select the group "Genes and Gene Prediction Tracks," the track "RefSeq Genes," and the table "RefSeq Genes." We select the option "All User Track records that have any overlap with RefSeq Genes." Click submit. Now on the table browser page select the output format such as "sequence" and "get output."

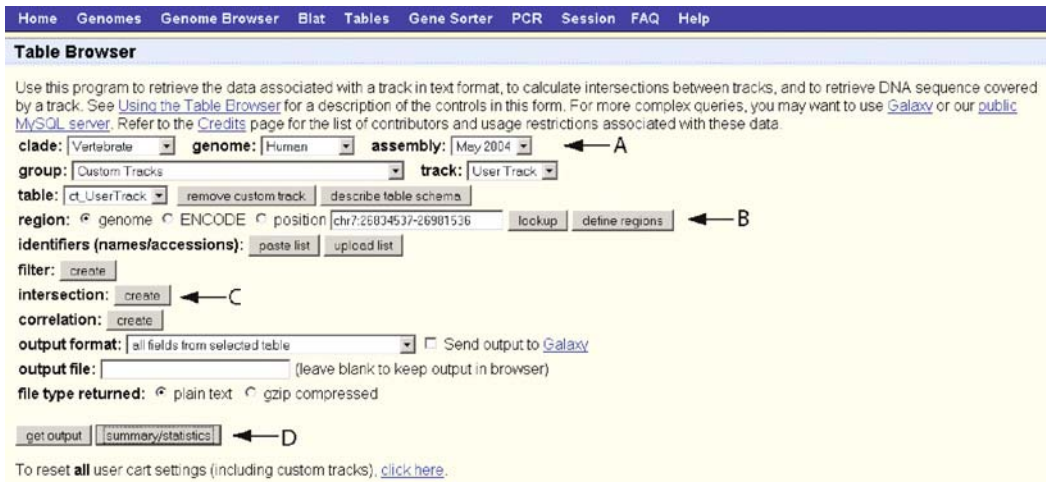


Fig. 14.6. The Table Browser allows a broad range of queries and data downloads. Here, the results of a BLAT search have been uploaded. Features include the choice of genome and assembly (*arrow* **A**); region of interest (*arrow* **B**); filters, intersections (*arrow* **C**) or correlations; output formats; and the option of getting a full output or a brief summary. Note that if an output file is specified, the result is downloaded to the local hard drive.

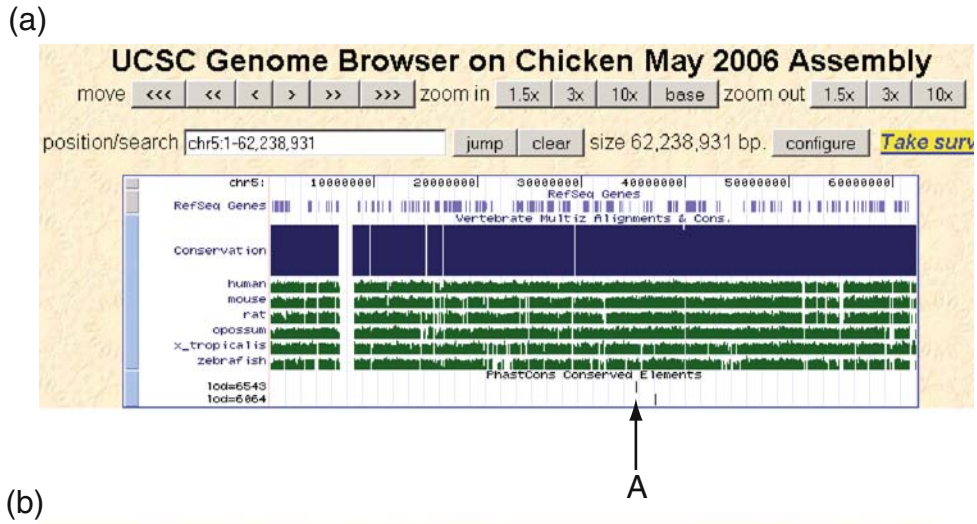
3.4. Highly Conserved Elements: From Chicken to Human

The comparison of vertebrate (and other) genomes has revealed the existence of highly conserved (or "ultraconserved") elements, many of which occur in non-coding regions *(15–17)*. We will use the Genome Browser and the Table Browser to explore several of these.

From the home page of the UCSC Genome Bioinformatics site, select the Genome Browser (or click the "Genomes" link). Set the clade to vertebrate and the genome to chicken (May 2006 assembly). Set the position to chr5, showing all of chromosome 5 (about 62 million base pairs). Note that the number of available annotation tracks is considerably fewer for chicken than for human.

In the Comparative Genomics category, set the "Conservation" track to "full." Then click the link for "Most Conserved." This resource shows conserved elements identified using the PhastCons program *(16)*. The scores range from 0 to 1,000; set the cutoff to 900, and set the view to full. There are two highly conserved elements on this chromosome (**Fig. 14.7a**).

View the element to the left (**Fig. 14.7a**, arrow A). You can zoom in threefold by clicking on the top line of the browser box (on the row indicated by chr5). By repeatedly clicking, the element is apparent in a window of about 10,000 base pairs (chr5:35,192,840–35,202,325) or ~3,000 base pairs (chr5:35,196,792–35,199,953). On the conservation track, click to view a multiple sequence alignment of these nucleotides from chicken, human, mouse, rat, opossum, *Xenopus tropicalis*, and zebrafish (**Fig. 14.7b**).



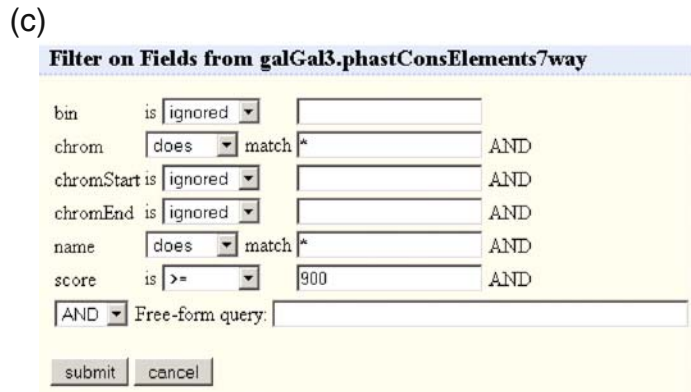


Fig. 14.7. Analysis of genomic regions that are highly conserved between chicken, human, and other vertebrates. (**a**) The Genome Browser display of chicken chromosome 5 includes two highly conserved elements as identified using the PhastCons program. (**b**) By zooming in on the element labeled with *arrow A* and then clicking the conservation block, a multiple sequence alignment is displayed. (**c**) Using the Table Browser, the identity and genomic coordinates of regions of interest can be listed. Here the PhastCons elements are filtered to analyze only those having a score $\geq$ 900.

Analyze highly conserved elements in chicken (and other species) using the UCSC Table Browser. Use the following settings: clade, vertebrate; genome, chicken; assembly, May 2006; group,

Comparative Genomics; track, Most Conserved; table, phastConsElements7way; region, genome. To further specify that we are interested in conserved elements having very high scores, at the filter setting click "create" and set the score ≥ 900 (**Fig. 14.7c**). Next, click the summary/statistics button to see that there are six elements matching our criteria. Do any of these occur inside a gene? Click "create" intersection, and set the group to "Genes and Gene Prediction Tracks," the track to "RefSeq Genes," and the table to "RefSeq Genes." The summary/statistics button shows that one element meets this criterion. You can set the output to a sequence file to study this further, or to a hyperlink to view it on the Chicken Genome Browser; the highly conserved gene is *DDX3X* (chr1:115,609,156–115,611,727), a DEAD box RNA helicase (NM_001030800).

3.5. Highly Conserved Elements: A Disease-Associated Region

Van Buchem (VB) disease is a severe sclerosing bone dysplasia. It is caused by the homozygous deletion of 52 kilobases of non-coding DNA downstream of the *SOST* gene on 17p21. (Patients with sclerosteosis have homozygous null mutations in *SOST* itself, but VB patients do not harbor mutations in the *SOST* gene.) We can use the Genome Browser to explore the conserved regulatory elements downstream from the gene. Understanding these conserved elements might lead to an understanding of the functional consequence of the disease-associated deletion.

Select the May 2004 build of the UCSC Human Genome Browser, and enter SOST. From the list of RefSeq genes, choose SOST on chromosome 17. Turn on the following tracks: Base Position (full), Chromosome Band (full), and RefSeq genes (pack). Adjust the window size to the coordinates 17:39,090,001–39,200,000. This is a range of 110,000 base pairs including the SOST gene and (to its left) the downstream MEOX1 gene. The 52-kilobase VB deletion is situated between these two genes.

The VB deletion region is flanked by two Alu repeats (a type of short interspersed nuclear element or SINE). To view Alu repeats in this region, go to the "Variation and Repeats" section and set RepeatMasker to full. Set the Conservation track (in the "Comparative Genomics" section) to full. A display is shown (**Fig. 14.8a**) (created using the PDF/PS option on the top bar of the website to download a postscript file). A group of peaks is evident, corresponding to highly conserved sites. Loots et al. *(18)* selected seven of these regions to test for enhancer activity. They found that one (ECR5) stimulated transcription of a heterologous promoter in osteopblastic but not kidney derived cell lines.

Identify the region of ECR5 and its relationship to the conserved regions displayed by the Genome Browser. To do this, rely on the fact that Loots et al. *(18)* specified the primers they used to amplify ECR5. The primers were

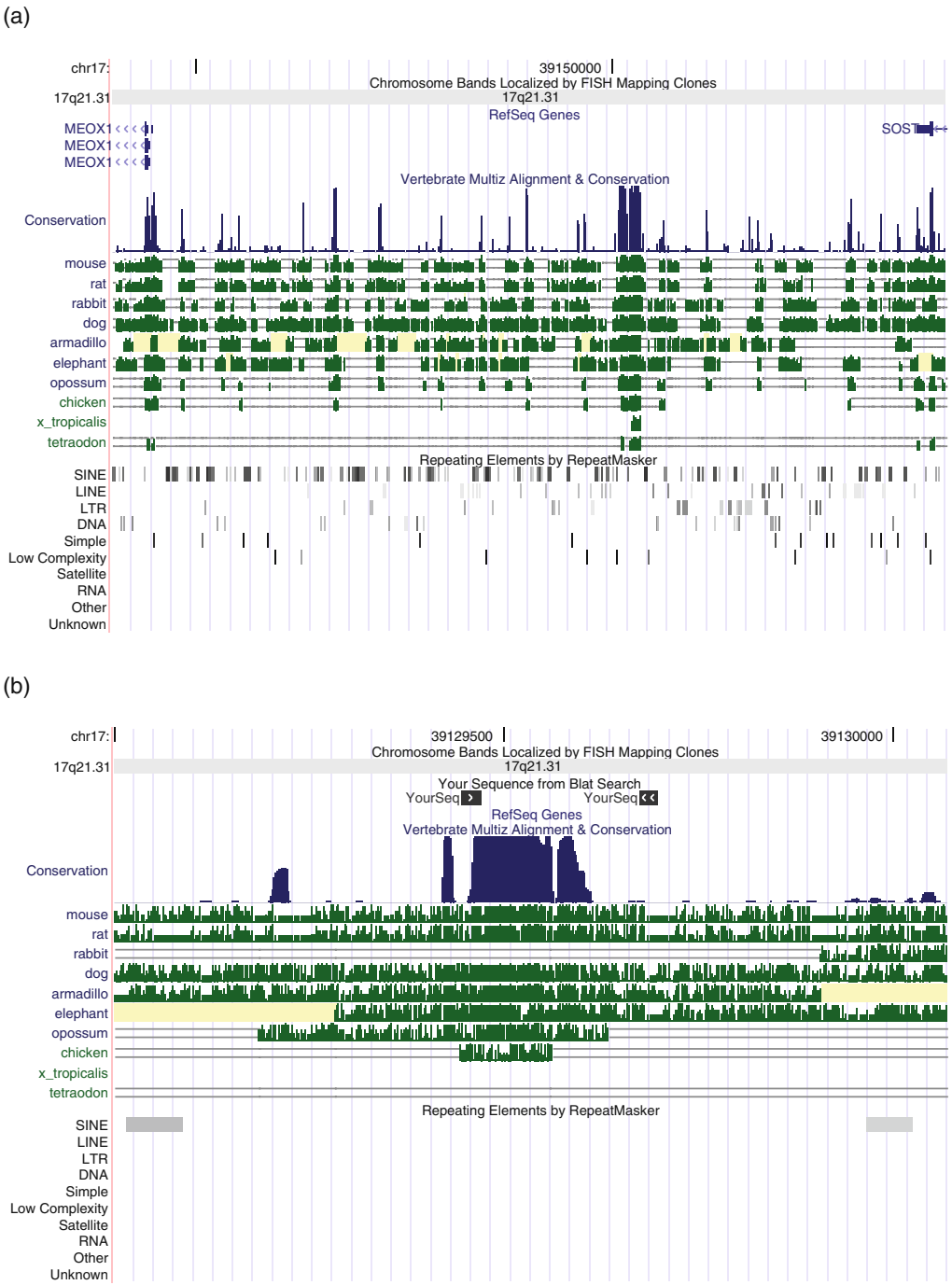


Fig. 14.8. Analysis of conserved elements in a disease-associated region. (**a**) View of the region on human chromosome 17 from the *SOST* gene (implicated in Van Buchem disease) to the downstream *MEOX* gene. Conservation tracks reveal highly genomic regions that are highly conserved across species, and RepeatMasker tracks reveal many classes of repetitive elements. The size of the region is 110,000 base pairs. (**b**) Detailed view of a region containing a conserved element with flanking SINEs. The positions of oligonucleotides used to amplify a portion of this conserved region are shown, based on a BLAT search.

5′-TCCTTGCCACGGGCCACCAGCTTT-3′ and 5′-CCCCCTCATGGCTGGTCTCATTTG-3′. Enter these oligonucleotide sequences (as a single string, without the 5′ and 3′ labels) into BLAT at the UCSC website. Choose one of the results on chromosome 17 (in the vicinity of SOST). Use the genome browser button to zoom out, or manually enter the coordinates chr17:39,129,000–39,130,070. This shows the region of both primers (labeled as "yourseq") (**Fig. 14.8b**).

Click on the blue peak of the conservation track. This displays a multiple sequence alignment of the evolutionarily conserved region. Inspecting the conservation track, it appears that this evolutionarily conserved region containing ECR5 (chr17:39,129,357–39,129,713) is present in the chicken genome, which diverged from humans over 300 million years ago.

3.6. Detailed Genome Analysis Via the ENCODE Project

The Encyclopedia of DNA Elements (ENCODE) project is designed to identify all the functional elements in the human genome *(19)*. In the pilot phase of the project, 44 ENCODE regions were selected that span 30 million base pairs (~1% of the human genome). ENCODE regions have been comprehensively sequenced in humans and in several dozen model organisms, making them ideal for comparative genomic analyses.

In the pilot phase of the ENCODE project, over 30 labs supplied data contributing to 56 browser tracks and 385 database tables *(20)*. The Genome Browser offers annotation tracks for ENCODE regions in six categories: (1) ENCODE regions and genes, (2) transcript levels, (3) chromatin immunoprecipitation, (4) chromosome, chromatin and DNA structure, (5) comparative genomics, and (6) variation. To illustrate the usefulness of the ENCODE regions, we will analyze aspects of the Hox gene cluster ENCODE region using the UCSC Genome Browser.

As part of the ENCODE project, several groups carefully assessed the transcriptional activity of ENCODE regions using laboratory-based approaches such as quantitative real-time polymerase chain reaction. This provided a benchmark with which to test the abilities of computational gene-finding programs. In the EGASP competition *(21)*, over a dozen research groups predicted genes and gene structures in ENCODE regions, using a combination of ab initio prediction and prior experimental evidence such as EST data. Subsequently, the sensitivity and specificity of the methods were assessed. The results of this competition are significant because they show which approaches are likely to have fewest false positives and false negatives when applied to the remainder of the human genome and other genomes.

From the UCSC Genome Browser home page, select ENCODE on the sidebar, then select "Regions (Hg17)." You can now select an ENCODE region from the left sidebar. Options include well-characterized loci (e.g., *CFTR*, interleukin, apo

cluster, alpha globin, beta globin, *FOXP2*) as well as a variety of manually and randomly selected regions. Click on ENm010 corresponding to the *HOXA* cluster.

The *HOXA* cluster on chromosome 7p is now displayed (with a selection of 0.5 megabases) with coordinates chr7:26,730,761–27,230,760. To further focus on these genes, change the coordinates to chr7:26,800,001–27,000,000. This shows a selection of 100,000 base pairs having no annotated genes, followed by 100,000 base pairs including the *HOXA* genes. Turn on the following three tracks: base position (dense), Known Genes (full), and ENCODE Regions (pack; this confirms that the browser view is in an ENCODE region).

In the "ENCODE Regions and Genes" category, click on the "EGASP Full" link. This allows you to read details about the EGASP project for gene prediction. Set the display mode to dense and click the submit button. Inspect the results on the genome browser: gene structures are presented for 15 different gene-finding algorithms (**Fig. 14.9**). Note that these algorithms

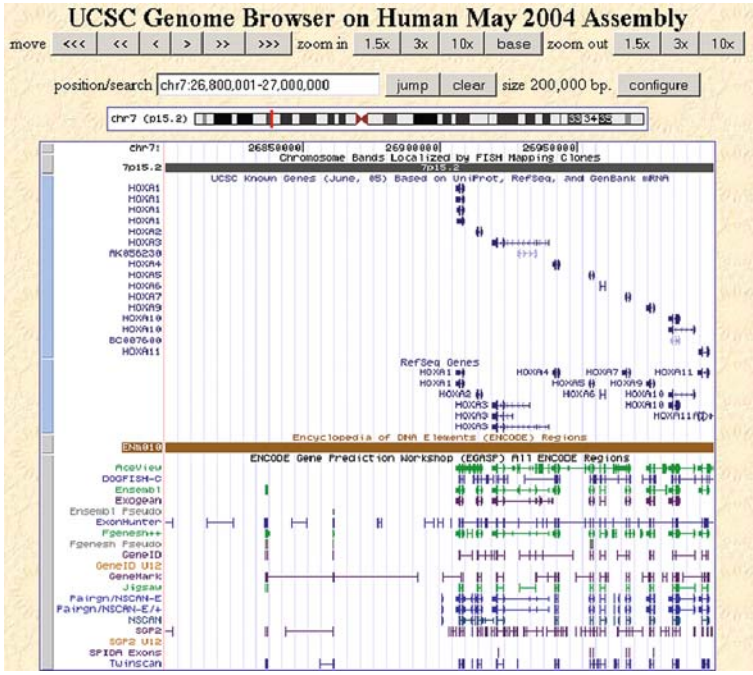


Fig. 14.9. Genome Browser view spanning part of the ENCODE region for *HOXA1*. The EGASP annotation track (set to pack) shows gene models predicted by a variety of computational gene-finding programs. Note that there is substantial overlap with known *HOX* genes, but additional exons and alternative gene structures are predicted at a variety of loci.

can differ dramatically in the gene structures they predict, including the number of exons, the length of exons, and the identity of the initial and terminal exons.

Inspect the left half of the display, in which there are no UCSC Known Genes but there are various predicted gene models. To view these 100,000 base pairs, set the coordinates to chr7:26,800,001–26,900,000. Might there be ESTs that provide evidence supporting the existence of genes in this region? Go to the "mRNA and EST Tracks" category and set human ESTs to pack. Indeed, there are many human ESTs in this region, although none have been incorporated into full gene models as part of the UCSC Known Gene track.

To list the ESTs in this region, click "Table" (on the top bar), set the group to "mRNA and EST Tracks," the track to "Human ESTs," the region to position chr7:26,800,001–26,900,000. Set the output to "all fields from selected table" then get the output. Eighteen ESTs in this region are listed. As an alternative way to view the data, set the output format to "sequence" and enter a title into the output file (e.g., "18_ests"). The 18 ESTs will be saved in the fasta format as part of a text document on the desktop.

3.7. Variation in the Human Genome

Genomic DNA is characterized by variation at many levels. SNPs are variants that occur between members of a population; in humans, over ten million SNPs have been identified. Other types of variation occur at a small scale, such as microsatellite and minisatellite sequences of varying lengths. Insertion and deletion (indel) polymorphisms also commonly occur. At larger scales, copy number variants have been reported both in the apparently normal population and in diseases such as autism. The UCSC Genome Browser offers a variety of tracks that display variation *(22)*.

From the home page of the UCSC Genome Bioinformatics site, select the ENCODE regions (hg17) and ENm010 (HOXA cluster on chromosome 7). The window size is 500,000 base pairs, and the coordinates are chr7:26,730,761–27,230,760. Set the base position to dense, and the RefSeq genes to "pack."

Go to the "Variation and Repeats" category. We begin with three of the tracks that show small-scale variation. (1) Set "RepeatMasker" to "full." RepeatMasker is a popular program that uses the RepBase library of repeats *(23)*. It is used to identify repetitive elements including short interspersed nuclear elements (SINEs), long interspersed nuclear elements (LINEs), long terminal repeats (LTRs), DNA transposons, simple repeats, low complexity regions, and satellite sequences. The main display window shows a variety of these repeating elements. SINE, LINE, LTR, and DNA repeats are noticeably absent from the cluster of HOXA

genes. (2) Set "Simple Repeats" to "pack." The simple repeats track shows the simple tandem repeats identified by the TRF program *(24)*. (3) Set "Microsatellite" to "full." This shows dinucleotide and trinucleotide patterns such as 17 consecutive AG dinucleotides.

3.8. Adding Custom Tracks to the Genome Browser

The UCSC Genome Browser offers the powerful feature of allowing you to upload custom tracks. This may be useful for a broad variety of applications such as

– viewing all the human genes that encode kinases;

– creating a track for chromosomal loci for which you have obtained gene expression data of interest;

– summarizing information about almost any large-scale experiment (such as those involving single nucleotide polymorphism arrays, array comparative genome hybridization, or methylation arrays) for which data have been obtained and chromosomal loci are available;

– we discussed the use of custom tracks from BLAT results in **Section 3.3** above.

Thus, custom tracks offer tremendous flexibility. Once they are created, they can be analyzed in the genome browser and/or the table browser. Detailed instructions for constructing and uploading custom tracks are available from the UCSC Genome Browser website.

You first need to format the data set in a tab-delimited text file. This can be done using Microsoft® Excel, Microsoft® Notepad, or an editor such as Crimson Editor (http://www.crimsoneditor.com/). The data may be in a variety of file formats (*see* **Note 5**). These include General Feature Format (GFF), Gene Transfer Format (GTF), PSL, BED, or WIG. For this example we use the BED format (**Fig. 14.10a**). Line 1, called a browser line, defines the initial browser position and in this case is set to match the ENCODE coordinates for the *HOXA1* cluster. Lines 4–7 define a custom track. The header (line 4) includes a name and a description that appear once this file is uploaded to the UCSC Genome Browser, a visibility indicator (with values 0 to hide; 1 for dense; 2 for full display; 3 for pack; and 4 for squish). The data (lines 5–7) include three mandatory fields containing (1) the chromosome (chr7 in this case), (2) start position, and (3) stop position. There are nine additional fields: (4) name (custom_block1 in this example), (5) score (including a browser option to limit output to scores above some user-defined threshold), (6) the plus or minus strand (note that both + and – strand options are used in this **Fig. 14.9**), (7) the thick line start position, (8) the thick line end position, and (9) the color (0,0,0 specifies black; in

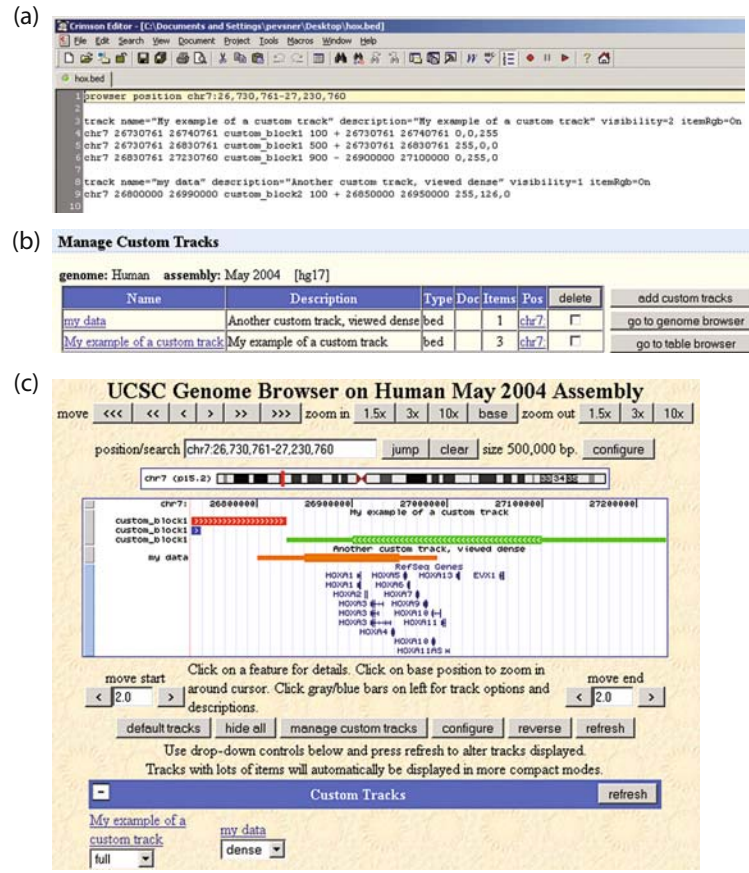


Fig. 14.10. Creating and viewing custom annotation tracks using BED files. (**a**) Screen capture of a BED formatted data file using the Crimson Editor text editor. (Other word or spreadsheet programs may also be used.) The information in this file includes the display properties (e.g., whether the track is to be shown in the full or dense viewing mode), the chromosome and start and end positions, names for the custom data sets, and the color and line thicknesses to be displayed. (**b**) Once a BED file is created, it can be uploaded to the UCSC Genome Browser site, allowing access to the Genome Browser or Genome Table. (**c**) The Genome Browser displays the data as specified in the BED file. Here, several colors are used as well as thick and thin lines (that can correspond to features within the track such as the presence of exons and introns or deletions and duplications).

Fig. 14.9a 0,0,255 is red; 255,0,0 is blue; 0,255,0 is green; 255,126,0 is orange). In this example we also show a second custom track labeled "my data" (**Fig. 114.0a**, lines 9–10). A BED file may include any number of blocks of data of interest.

From the main page of the UCSC Genome Browser, select the "add custom tracks" button. There is an option to paste URLs or data, or to upload a file; in this example, upload a file

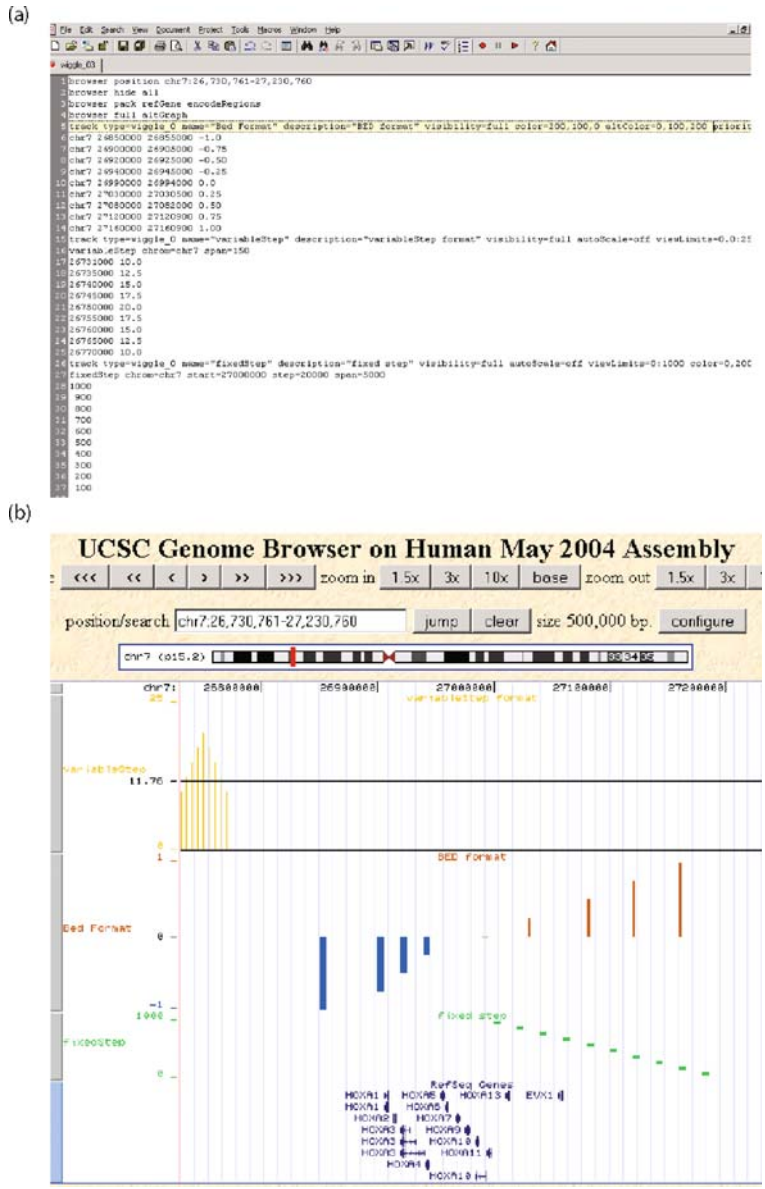


Fig. 14.11. Creating and viewing custom annotation tracks using WIG files. The WIG output is useful for continuous data and offers more sophisticated output properties than a BED file. (**a**) Example of a WIG file (shown in Crimson Editor) with three different data sets. (**b**) View of the custom annotation tracks in the UCSC Genome Browser. Note the different types of graphical display. Many features are modifiable by the user of WIG tracks, such as the axis heights and labeling.

with the extension ".bed" by using the browse then submit buttons. You can now click the link "go to genome browser" (**Fig. 14.10b**).

View the custom tracks. In this case there are four custom tracks (one set in red, blue, and green; another track in orange) with features specified in the BED file. There is also a custom tracks category at the beginning of the annotation tracks section (**Fig. 14.10c**, bottom); this allows you to change the viewing properties of the tracks or to obtain further information about the nature of the custom tracks.

Custom tracks can be created with data in the WIG (wiggle) format. This is particularly useful for the display of continuous-valued data across tracks. An example is shown in **Fig. 14.11**. A wig-formatted file has a track definition line (which controls the display options) and then track data. These are usually entered using the fixedStep data type for single column data or the variableStep data type for two column data. It is also possible to use the BED format, discussed above, in a wiggle file. Examples of each of these data types are shown in **Fig. 14.10a** and the browser output is shown in **Fig. 14.10b**.

4. Notes

1. You can select different assemblies for various organisms. For example, you can specify May 2006 instead of March 2004 for human. Each assembly has unique annotations and in some cases later assemblies have fewer annotations. It is possible to relate the genomic coordinates in two assemblies by using the Convert tool (accessible from the top bar of a Genome Browser view).
2. The RefSeq project at NCBI is designed to provide a curated, representative accession number for DNA and protein sequences. Many DNA and protein sequences have accession numbers assigned to various research groups and/or to sequence variants. As an example, there are over 250,000 accession numbers assigned to human immunodeficiency virus-1 variants. However, there is only a single RefSeq accession number assigned to HIV-1 DNA and protein sequences. In general, the RefSeq accession should be used whenever possible. The format of a RefSeq protein is in a format such as NP_123456 or XP_123456, while mRNA-based sequences have the format NM_123456 or XM_123456. Non-RefSeq accessions may have a variety of formats such as J123456 or P654321. If you use the UCSC Known Genes track, you will obtain a broader coverage of information about genes, but it may be less well annotated than RefSeq.

3. The web-based BLAT program restricts the query to 25,000 nucleotides. For longer queries, you can download the BLAT executables.
4. The fasta format consists of an optional line of text beginning with the > sign followed by a single line break, then the DNA or protein sequence in single letter abbreviation code, without line breaks. Entries at the NCBI Entrez Protein or Entrez Nucleotide databases can be displayed in the fasta format.
5. File formats for creating custom tracks include General Feature Format (GFF), Gene Transfer Format (GTF), Pat space layout (PSL), Blue Elephant Definition (BED), or Wiggle Format (WIG). These vary in the types of features they offer, such as the ability to include line plots. A variety of publicly available custom tracks (listed at http://genome.ucsc.edu/goldenPath/customTracks/custTracks.html) are available in these various formats.

Acknowledgments

I thank members of my lab (Erica Aronson, Laurence Frelin, Nathaniel Miller, Elisha Roberson, Asha Shah, Jason Ting) for helpful discussions, and N. Varg for comments on the manuscript. This work was supported in part by NIH grants HD046598 and 24061HD.

References

1. Furey, T. S. (2006) Comparison of human (and other) genome browsers. *Hum Genomics* **2**, 266–70.
2. Kuhn, R. M., Karolchik, D., Zweig, A. S., Trumbower, H., Thomas, D. J., Thakkapallayil, A., Sugnet, C. W., Stanke, M., Smith, K. E., Siepel, A., Rosenbloom, K. R., Rhead, B., Raney, B. J., Pohl, A., Pedersen, J. S., Hsu, F., Hinrichs, A. S., Harte, R. A., Diekhans, M., Clawson, H., Bejerano, G., Barber, G. P., Baertsch, R., Haussler, D., and Kent, W. J. (2007) The UCSC genome browser database: update 2007. *Nucleic Acids Res* **35**(Database issue), D668–73.
3. Wheeler, D. L., Barrett, T., Benson, D. A., Bryant, S. H., Canese, K., Chetvernin, V., Church, D. M., DiCuccio, M., Edgar, R., Federhen, S., Geer, L. Y., Kapustin, Y., Khovayko, O., Landsman, D., Lipman, D. J., Madden, T. L., Maglott, D. R., Ostell, J., Miller, V., Pruitt, K. D., Schuler, G. D., Sequeira, E., Sherry, S. T., Sirotkin, K., Souvorov, A., Starchenko, G., Tatusov, R. L., Tatusova, T. A., Wagner, L., and Yaschenko, E. (2007) Database resources of the National Center for Biotechnology Information. *Nucleic Acids Res* **35**(Database issue), D5–12.
4. Birney, E., Andrews, D., Caccamo, M., Chen, Y., Clarke, L., Coates, G., Cox, T., Cunningham, F., Curwen, V., Cutts, T., Down, T., Durbin, R., Fernandez-Suarez, X. M., Flicek, P., Graf, S., Hammond, M., Herrero, J., Howe, K., Iyer, V., Jekosch, K., Kahari, A., Kasprzyk, A., Keefe, D., Kokocinski, F., Kulesha, E., London, D., Longden, I., Melsopp, C., Meidl, P., Overduin, B., Parker, A., Proctor, G., Prlic, A., Rae, M., Rios, D., Redmond, S., Schuster, M., Sealy, I., Searle, S., Severin, J., Slater, G., Smedley, D., Smith, J., Stabenau, A., Stalker, J.,

Trevanion, S., Ureta-Vidal, A., Vogel, J., White, S., Woodwark, C., and Hubbard, T. J. (2006) Ensembl 2006. *Nucleic Acids Res* **34**(Database issue), D556–61.

5. Hubbard, T. J., Aken, B. L., Beal, K., Ballester, B., Caccamo, M., Chen, Y., Clarke, L., Coates, G., Cunningham, F., Cutts, T., Down, T., Dyer, S. C., Fitzgerald, S., Fernandez-Banet, J., Graf, S., Haider, S., Hammond, M., Herrero, J., Holland, R., Howe, K., Howe, K., Johnson, N., Kahari, A., Keefe, D., Kokocinski, F., Kulesha, E., Lawson, D., Longden, I., Melsopp, C., Megy, K., Meidl, P., Ouverdin, B., Parker, A., Prlic, A., Rice, S., Rios, D., Schuster, M., Sealy, I., Severin, J., Slater, G., Smedley, D., Spudich, G., Trevanion, S., Vilella, A., Vogel, J., White, S., Wood, M., Cox, T., Curwen, V., Durbin, R., Fernandez-Suarez, X. M., Flicek, P., Kasprzyk, A., Proctor, G., Searle, S., Smith, J., Ureta-Vidal, A., and Birney, E. (2007) Ensembl 2007. *Nucleic Acids Res* **35**(Database issue), D610–17.
6. Hinrichs, A. S., Karolchik, D., Baertsch, R., Barber, G. P., Bejerano, G., Clawson, H., Diekhans, M., Furey, T. S., Harte, R. A., Hsu, F., Hillman-Jackson, J., Kuhn, R. M., Pedersen, J. S., Pohl, A., Raney, B. J., Rosenbloom, K. R., Siepel, A., Smith, K. E., Sugnet, C. W., Sultan-Qurraie, A., Thomas, D. J., Trumbower, H., Weber, R. J., Weirauch, M., Zweig, A. S., Haussler, D., and Kent, W. J. (2006) The UCSC genome browser database: update 2006. *Nucleic Acids Res* **34**(Database issue), D590–98.
7. Karolchik, D., Hinrichs, A. S., Furey, T. S., Roskin, K. M., Sugnet, C. W., Haussler, D., and Kent, W. J. (2004) The UCSC table browser data retrieval tool. *Nucleic Acids Res* **32**(Database issue), D493–6.
8. Hoegg, S., and Meyer, A. (2005) Hox clusters as models for vertebrate genome evolution. *Trends Genet* **21**(8):421–4.
9. Hsu, F., Kent, W. J., Clawson, H., Kuhn, R. M., Diekhans, M., and Haussler, D. (2006) The UCSC known genes. *Bioinformatics* **22**(9):1036–46.
10. The UniProt Consortium (2007) The Universal Protein Resource (UniProt). *Nucleic Acids Res* **35**(Database issue), D193–7.
11. Benson, D. A., Karsch-Mizrachi, I., Lipman, D. J., Ostell, J., and Wheeler, D. L. (2007) GenBank. *Nucleic Acids Res* **35**(Database issue), D21–5.
12. Altschul, S. F., Gish, W., Miller, W., Myers, E. W., and Lipman, D. J. (1990) Basic local alignment search tool. *J Mol Biol* **215,** 403–10.
13. Altschul, S. F., Madden, T. L., Schaffer, A. A., Zhang, J., Zhang, Z., Miller, W., and Lipman, D. J. (1997) Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Res* **25,** 3389–402.
14. Kent, W. J. (2002) BLAT – the BLAST-like alignment tool. *Genome Res* **12,** 656–64.
15. Bejerano, G., Pheasant, M., Makunin, I., Stephen, S., Kent, W. J., Mattick, J. S., and Haussler, D. (2004) Ultraconserved elements in the human genome. *Science* **304**(5675):1321–5.
16. Siepel, A., Bejerano, G., Pedersen, J. S., Hinrichs, A., Hou, M., Rosenbloom, K., Clawson, H., Spieth, J., Hillier, L. W., Richards, S., Weinstock, G. M., Wilson, R. K., Gibbs, R. A., Kent, W. J., Miller, W., and Haussler, D. (2005) Evolutionarily conserved elements in vertebrate, insect, worm, and yeast genomes. *Genome Res* **15,** 1034–50.
17. Sun, H., Skogerbo, G., and Chen, R. (2006) Conserved distances between vertebrate highly conserved elements. *Hum Mol Genet* **15,** 2911–22.
18. Loots, G. G., Kneissel, M., Keller, H., Baptist, M., Chang, J., Collette, N. M., Ovcharenko, D., Plajzer-Frick, I., and Rubin, E. M. (2005) Genomic deletion of a long-range bone enhancer misregulates sclerostin in Van Buchem disease. *Genome Res* **15,** 928–35.
19. ENCODE Project Consortium (2004) The ENCODE (ENCyclopedia Of DNA Elements) Project. *Science* **306,** 636–40.
20. Thomas, D. J., Rosenbloom, K. R., Clawson, H., Hinrichs, A. S., Trumbower, H., Raney, B. J., Karolchik, D., Barber, G. P., Harte, R. A., Hillman-Jackson, J., Kuhn, R. M., Rhead, B. L., Smith, K. E., Thakkapallayil, A., Zweig, A. S., ENCODE Project Consortium, Haussler, D., and Kent, W. J. (2007) The ENCODE Project at UC Santa Cruz. *Nucleic Acids Res* **35**(Database issue), D663–67.
21. Guigo, R., Flicek, P., Abril, J. F., Reymond, A., Lagarde, J., Denoeud, F., Antonarakis, S., Ashburner, M., Bajic, V. B., Birney, E., Castelo, R., Eyras, E., Ucla, C., Gingeras, T. R., Harrow, J., Hubbard, T., Lewis, S. E., Reese, M. G. (2006) EGASP: the human ENCODE Genome Annotation

Assessment Project. *Genome Biol* **7** Suppl 1:S2.1–31.

22. Thomas, D. J., Trumbower, H., Kern, A. D., Rhead, B. L., Kuhn, R. M., Haussler, D., and Kent, W. J. (2007) Variation resources at UC Santa Cruz. *Nucleic Acids Res* **35**(Database issue), D716–20.
23. Jurka, J. (2000) Repbase update: a database and an electronic journal of repetitive elements. *Trends Genet* **16**(9):418–20.
24. Benson, G. (1999) Tandem repeats finder: a program to analyze DNA sequences. *Nucleic Acids Res* **27**(2), 573–80.

Chapter 15

Mining for SNPs and SSRs Using SNPServer, dbSNP and SSR Taxonomy Tree

Jacqueline Batley and David Edwards

Abstract

Molecular genetic markers represent one of the most powerful tools for the analysis of genomes and the association of heritable traits with underlying genetic variation. The development of high-throughput methods for the detection of single nucleotide polymorphisms (SNPs) and simple sequence repeats (SSRs) has led to a revolution in their use as molecular markers. The availability of large sequence data sets permits mining for these molecular markers, which may then be used for applications such as genetic trait mapping, diversity analysis and marker assisted selection in agriculture. Here we describe web-based automated methods for the discovery of SSRs using SSR taxonomy tree, the discovery of SNPs from sequence data using SNPServer and the identification of validated SNPs from within the dbSNP database. SSR taxonomy tree identifies pre-determined SSR amplification primers for virtually all species represented within the GenBank database. SNPServer uses a redundancy based approach to identify SNPs within DNA sequences. Following submission of a sequence of interest, SNPServer uses BLAST to identify similar sequences, CAP3 to cluster and assemble these sequences and then the SNP discovery software autoSNP to detect SNPs and insertion/deletion (indel) polymorphisms. The NCBI dbSNP database is a catalogue of molecular variation, hosting validated SNPs for several species within a public-domain archive.

Key words: Single nucleotide polymorphism, SNP, SNPServer, autoSNP, dbSNP, simple sequence repeat, SSR, SSR taxonomy tree, SSRPrimer.

1. Introduction

Molecular marker technology has developed rapidly over the last decade, and two forms of sequence based marker, single nucleotide polymorphisms (SNPs) and simple sequence repeats (SSRs) (*see* **Note 1**), now predominate applications in modern genetic analysis. SNPs and SSRs have many uses in human genetics, such as the detection of alleles associated with genetic diseases, paternity

David Posada (ed.), *Bioinformatics for DNA Sequence Analysis,* Methods in Molecular Biology 537

DOI 10.1007/978-1-59745-251-9_15

assessment, forensics and inferences of population history *(1, 2)*. Furthermore, these markers are invaluable as a tool for genome mapping in all systems, offering the potential for generating very high density genetic maps that can be used to develop haplotypes for genes or regions of interest *(3)*. The application of molecular markers to advance plant and animal breeding is now well established *(4)*. Modern agricultural breeding is dependent on molecular markers, from trait mapping to marker assisted selection. Molecular markers can be used to select parental genotypes in breeding programs, eliminate linkage drag in back-crossing and select for traits that are difficult to characterise phenotypically.

1.1. SNPs Molecular Genetic Markers

DNA sequence differences are the basic requirement for the study of molecular genetics. A SNP is an individual nucleotide base difference between two DNA sequences. SNPs are categorised as either transitions (C/T or G/A), transversions (C/G, A/T, C/A, or T/G) or small insertions/deletions (indels). SNPs are the ultimate form of molecular genetic marker, as a nucleotide base is the smallest unit of inheritance. SNPs are direct markers because the sequence information provides the exact nature of the allelic variants and this sequence variation can have a major impact on how the organism develops and responds to the environment. Furthermore, SNPs may provide a high density of markers near a locus of interest as they represent the most frequent type of genetic polymorphism. Recent evidence has shown that when comparing human DNA from two individuals, SNPs are found on average every 1–2 kb *(5, 6)*. Studies of sequence diversity have now been performed for a range of plant species and these have indicated that SNPs appear to be even more abundant in plant systems than in the human genome, with 1 SNP every 100–300 bp *(7)*. SNPs at any particular site could in principle involve four different nucleotide variants, but in practice they are generally biallelic. This disadvantage, when compared with multiallelic markers such as SSRs, is compensated by the relative abundance of SNPs. SNPs are also evolutionarily stable, not changing significantly from generation to generation. This low mutation rate of SNPs makes them excellent markers for studying complex genetic traits and as a tool for the understanding of genome evolution *(8)*.

SNPs provide an important source of molecular markers and are suitable for automated discovery and detection for many applications. These markers are invaluable as a tool for genome mapping, offering the potential for generating ultra-high density genetic maps that can be used to develop haplotyping systems for genes or regions of interest, map-based positional cloning, QTL detection and the assessment of genetic relationships between individuals. The international HapMap project *(9)* endeavours to characterise human genetic variation at the SNP level, adding significant value to the human genome sequence. SNPs are used

routinely in agriculture as markers in crop and livestock breeding programs *(4)*, for example, for genetic diversity analysis, cultivar identification, phylogenetic analysis, characterisation of genetic resources and association with agronomic traits *(3)*. The applications of SNPs in crop genetics have been extensively reviewed by Rafalski *(3)* and Gupta et al. *(4)*. These reviews highlight that for several years SNPs will co-exist with other marker systems and the use of SNPs will become more widespread with the increasing levels of genome sequencing, and the drop in price of SNP assays.

1.2. SNP Discovery from Sequence Data

As with the majority of molecular markers, one of the limitations of SNPs is the initial cost associated with their development. A variety of approaches have been adopted for the discovery of novel SNP markers. These fall into three general categories: in vitro discovery, where new sequence data is generated, in silico methods that rely on the analysis of available sequence data and indirect discovery, where the base sequence of the polymorphism remains unknown.

With the development of high-throughput sequencing technology, large amounts of data have been submitted to the various DNA databases that may be suitable for data mining and SNP discovery. This mining of sequence data sets should provide the cheapest source of abundant SNPs *(10–13)*. Expressed sequence tag (EST) sequencing programs have provided an enormity of information *(14)*, and this data may provide the richest source of biologically useful SNPs due to the relatively high redundancy of gene sequence, the diversity of genotypes represented within databases and the fact that each SNP would be associated with an expressed gene *(12)*. The continuing decrease in the cost of DNA sequencing is also leading to a growing number of whole genome sequencing projects. This data increasingly enables the identification of SNPs in overlapping genomic sequence and through comparison of the genomic sequences with EST sequences *(13, 15)*.

The development of high throughput methods for the detection of SNPs from bulk sequence data has led to a revolution in their use as molecular markers. However, although sequencing technologies continue to advance, high throughput sequencing remains prone to inaccuracies as frequent as one base in every hundred. This false base-calling can hamper the electronic filtering of sequence data to identify potentially biologically relevant polymorphisms. Therefore, the challenge of in silico SNP discovery is not the identification of SNPs, but the differentiation of true polymorphisms from the often more abundant sequence errors. There are several different sources of error which must be considered when differentiating between sequence errors and true SNPs. The automated reading of raw sequence data is the main source of sequencing error as there is a requirement to gain as long a sequence as possible with confidence in correct base calling. Phred is the most widely adopted software to call bases from

chromatogram data *(16, 17)*. Phred has the benefit of providing a statistical estimate of the accuracy of calling each base and therefore provides a primary level of confidence that any sequence difference detected represents true genetic variation. There are several software packages which utilise this feature to estimate the confidence of sequence polymorphisms within alignments. For example, PolyBayes and PolyPhred can differentiate between true SNPs and sequence error where sequence trace files are available for comparison to filter out polymorphisms in traces of dubious quality (*see* **Note 2**). Unfortunately, complete sequence trace file archives are rarely available for large sequence datasets collated from a variety of sources. Furthermore, sequence quality based SNP discovery does not identify errors in sequences that were incorporated prior to the base calling process. The principle cause of these prior errors is the inherently high error rate of the reverse transcription process required for the generation of cDNA libraries for EST sequencing. Similar errors can also occur to a lesser extent in any PCR amplification process that may be part of a sequencing protocol. In situations where trace files are unavailable, the identification of sequence errors can be based on two further methods to determine SNP confidence: redundancy of the polymorphism in an alignment and co-segregation of SNPs with haplotype.

There are two methods currently available which apply both redundancy and haplotype co-segregation measures in the identification of SNPs: autoSNP *(18, 19)* and SNPServer *(20)*. The frequency of occurrence of a polymorphism at a particular locus provides a measure of confidence that the SNP represents a true polymorphism and is referred to as the SNP redundancy score. By including only SNPs that have a redundancy score of two or greater (two or more of the aligned sequences represent the polymorphism), almost all of the sequencing errors are removed. Although some true genetic variation is also ignored due to its presence only once within an alignment, the high degree of redundancy within the data permits the rapid identification of large numbers of SNPs without the requirement for sequence trace files. However, while redundancy based methods for SNP discovery are highly efficient, the non-random nature of sequence error may lead to certain sequence errors being repeated between runs due to conserved, complex DNA structures. Therefore, errors at these loci would have a relatively high SNP redundancy score and appear as confident SNPs. A co-segregation score based on whether a SNP position contributes to defining a haplotype is a further independent measure of SNP confidence. While sequencing errors may occur with regularity at certain positions within a sequencing read due to conserved sequence complexity, the probability of these errors being repeated between sequence reads

remains random. True SNPs that represent divergence between homologous genes co-segregate to define a conserved haplotype, whereas sequence errors do not co-segregate with haplotype.

Within autoSNP and SNPServer, the SNP redundancy score and co-segregation score together provide a valuable means for estimating confidence in the validity of SNPs within aligned sequences independent of sequence trace files or the source of the sequence error. Sequences may be submitted for assembly with CAP3 *(21)* or submitted preassembled in ACE format. Alternatively, a single sequence may be submitted for BLAST comparison *(22)* with a DNA sequence database. Identified sequences are then processed for assembly with CAP3 and subsequent redundancy based SNP discovery. SNPServer has an advantage in being the only real time web-based software, which allows users to rapidly identify SNPs in sequences of interest using public data.

1.3. The NCBI dbSNP Database

Where extensive research has been performed in SNP identification and validation, databases of validated polymorphisms provide a valuable resource for molecular genetic research. The primary SNP repository is the NCBI database dbSNP *(23, 24)*. This database was established to serve as a central repository for SNPs and short insertion/deletion polymorphisms (indels). These validated polymorphisms may be used by many additional laboratories, using the sequence information around the polymorphism, the specified experimental conditions and the detailed annotation of the SNP.

1.4. SSR Molecular Genetic Markers

Simple sequence repeats, also known as microsatellites, have been shown to be one of the most powerful genetic markers in biology. They are short stretches of DNA sequence occurring as tandem repeats of mono-, di-, tri-, tetra-, penta- and hexa-nucleotides, widely and ubiquitously distributed throughout eukaryotic genomes *(25, 26)*. Furthermore, SSRs have been found in all prokaryotic and eukaryotic genomes analysed to date *(27)*. SSRs are highly polymorphic and informative markers and provide a valuable tool for characterising germplasm. The value of SSRs is due to their properties of genetic co-dominance, abundance, dispersal throughout the genome, multi-allelic variation, high reproducibility and amenability to automated allele detection and sizing *(28)*. The nature of SSRs gives them a number of advantages over other molecular markers: (i) multiple SSR alleles may be detected at a single locus using a simple PCR based screen, (ii) very small quantities of DNA are required for screening and (iii) analysis may be semi-automated. SSRs are highly polymorphic, due to mutation affecting the number of repeat units. This hypervariability makes them informative markers for a wide range of applications including high-density genetic mapping, molecular tagging of genes, genotype identification, analysis of genetic diversity, paternity exclusion, phenotype mapping and marker assisted selection for animal and

crop breeding *(29, 30)*. Furthermore, SSRs demonstrate a high degree of transferability between species, as PCR primers designed to an SSR within one species frequently amplify a corresponding locus in related species, making them excellent markers for comparative genetic and genomic analysis.

1.5. SSR Discovery from Sequence Data

The discovery and development of SSR molecular markers has traditionally been limited by the requirement to construct genomic DNA libraries enriched for SSR sequences and the subsequent specific DNA sequencing of these libraries *(31)*. This production of enriched libraries is time consuming and labour intensive and the specific sequencing required is expensive. Recent advances in bioinformatics have enabled the rapid discovery of SSR markers from bulk sequence data *(32)*. Therefore, as with SNPs, where abundant sequence data is already available, it is more economical and efficient to use computational tools to identify SSR loci. These markers are inexpensive to identify and can frequently be associated with functionally annotated genes.

By analysing the flanking DNA sequence, of the SSRs, for the presence of suitable forward and reverse PCR primers to assay the SSR loci, the discovery of SSRs can become an extremely rapid and inexpensive process. Several computational tools are currently available for the identification of SSRs within sequence data as well as for the design of PCR primers suitable for the amplification of specific loci. SSRPrimer *(32)* integrates two such tools, enabling the simultaneous discovery of SSRs within sequence data with the design of specific PCR primers for the amplification of these SSR loci. The web-based version of SSRPrimer permits the remote use of this tool with any sequence of interest. A further tool, SSR taxonomy tree, is described which demonstrates the application of SSRPrimer to the complete GenBank database, with the results organised as a taxonomic hierarchy for browsing or searching for SSR amplification primers for any species of interest *(33)*.

2. Program Usage

The SNPServer and SSR taxonomy tree web-based tools are hosted on linux and UNIX architecture ranging from a single processor linux server to a 64-node linux cluster and can be accessed through a standard web browser (http://bioinformatics.pbcbasc.latrobe.edu.au/index.html or http://acpfg.imb.uq.edu.au/). The tools are reliant on public software maintained on the host server, including BLAST *(22)*, CAP3 *(21)*, SPUTNIK *(34)*, PRIMER3 *(35)*, and custom scripts coded in Perl and Bioperl *(36)*.

2.1. SNPServer

SNPServer is based on the command line SNP discovery tool autoSNP and can be run either as a command line tool or web server. The web version of SNPServer provides an interface and wrapper for the three programs, BLAST, CAP3 and autoSNP, which make up the SNP discovery pipeline (**Fig. 15.1**). HTML format files are generated to allow the user to input sequence data as well as parameters for comparison, assembly and SNP discovery. The SNPServer pipeline accepts single sequences, a list of sequences in FASTA format for assembly or a pre-calculated sequence assembly in ACE format. Individual FASTA sequences are compared with a specified nucleotide sequence database using BLAST to identify related sequences, any number of which may then be selected for assembly with CAP3 and subsequent SNP discovery using autoSNP. The user can specify options for BLAST sequence comparison, CAP3 assembly and SNP discovery through the web interface. SNP discovery is performed using the redundancy-based approach described above, with a modified version of the autoSNP PERL script. The alignment data generated by CAP3, or from a user submitted ACE file, is used to load the sequences in each assembly into a 2D array. The identification of indels between sequences is permitted through the addition of spacing characters (•) added during sequence alignment, as a fifth element in addition to the four nucleotides A, C, G and T. Each row of the alignment represents a single-base locus in the assembly and is assessed for differing nucleotides. The minimum redundancy scores can be specified by the user and these are associated with the alignment width (the number of sequences included in the contig) to determine the number of different nucleotides at a base position required for classification as a SNP. Where a SNP is identified, the SNP score provided is equal to the minimum number of reads that share a common polymorphism. If more than one SNP is present in an alignment, a co-segregation score is calculated for each SNP. This is a measure of whether the SNP contributes to defining a haplotype. This score is then normalised to the number of sequences at that SNP position in the alignment, to produce a weighted co-segregation score. Access to

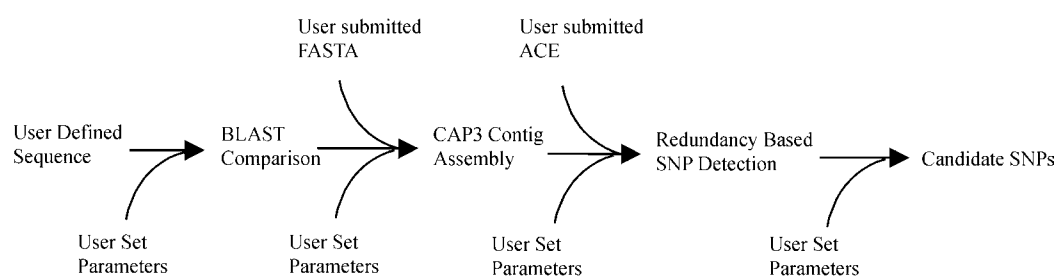


Fig. 15.1. SNPServer process pipeline.

the web server version requires an internet connection and a standard web browser. The host web server requires BLAST as well as a selection of BLAST-formatted databases: CAP3, Bioperl and the custom SNPServer perl scripts. Parameters for BLAST and CAP3 can be selected via the web interface and users are encouraged to assess different parameters as optimal parameters are strongly affected by the haplotype complexity of the gene of interest (*see* **Note 3**).

2.2. dbSNP

The Single Nucleotide Polymorphism database (dbSNP) was established by NCBI, as a public-domain archive for a broad collection of simple genetic polymorphisms. The database is a general catalogue of genetic variation for use in large scale projects such as association studies, genome mapping and evolutionary biology. The submissions to dbSNP are from a variety of sources such as individual groups, SNP discovery consortia, private industry and large-scale genome sequencing centres. Users may access dbSNP by three approaches: by sequence comparison or by SNP id at the dbSNP website; through Entrez; or through Locuslink. Here we will describe a search for a SNP using a known SNP id, as well as using the BLAST function to identify a SNP in a sequence of interest. dbSNP is stored at the NCBI in a MicroSoft SQL Server database. Access to the web server requires an internet connection and a standard web browser.

2.3. SSRPrimer and SSR Taxonomy Tree

SSRPrimer is an application which integrates SPUTNIK *(34)*, an SSR repeat finder, with Primer3 *(35)*, a PCR primer design program, into one pipeline tool *(32)*. The SSRPrimer tool has been applied to the complete GenBank database, resulting in the design of PCR amplification primers for 14 million SSRs, from a very broad range of species. The SSR taxonomy tree tool provides web-based searching and browsing of species and taxa for the visualisation and download of SSR amplification primers. There are two methods to access the SSR taxonomy tree. Users may enter a search string of either a common name or taxonomic key to identify a species of interest. Alternatively, the server permits browsing between related taxa. The SSRs and their amplification primers can be viewed or downloaded for specific species or taxonomic groups. SSRPrimer is a web-based tool that may also be run on the command line. All SSR PCR primers maintained within the SSR taxonomy tree were identified using SSRPrimer with the following parameters. Primer3 options are default with the following exceptions selected to increase primer specificity. One set of primer pairs are designed at least 10 bp distant from either side of the identified SSR. Optimum size for the primers are 21 bases with a maximum of 23 bases. Optimum melting temperature is 55°C with a minimum of 50°C and maximum of 70°C. The maximum

GC content is set to 70%. While these options may be modified on the SSRPrimer submission page, the authors suggest maintaining these strict criteria to ensure robust PCR amplification.

3. Examples

3.1. SNPServer: Identification of SNPs Within a Wheat EST

To begin, obtain the sequence of AL831313 from GenBank at NCBI (http://www.ncbi.nlm.nih.gov/). This is a wheat EST sequence produced as part of the BBSRC funded integrated gene function project *(37)*. Type AL831313 into the search bar, select the database as 'Nucleotide' from the drop down menu and click on 'GO'. This should return a single result. Display the sequence as FASTA format by selecting FASTA from the display drop down menu. Select and copy the resulting FASTA format sequence to the clipboard.

Open the SNPServer web page (http://hornbill.cspp.latrobe.edu.au/cgi-binpub/autosnip/index_autosnip.pl) and select BLAST (*see* **Note 4)**. Paste the FASTA format sequence into the box, select library 'Wheat' from the drop down menu, leaving the other parameters as default and click on 'Blast away'. This will provide a list of matching sequences when the BLAST is complete. Select 'Run SNP detection on all sequences' (*see* **Note 5**), this leads to a page which allows the user to select CAP3 assembly parameters and SNP discovery parameters (**Fig. 15.2**). Use the default parameters for routine SNP discovery and click on 'Start' (*see* **Note 3**).

A summary page is displayed upon the completion of the sequence assembly and SNP discovery (**Fig. 15.3**). In this example, all of the 29 sequences identified in the BLAST query assemble together. If CAP3 was unable to assemble any of the sequences they would be discarded. Information on the number of SNPs, SNP frequency, the cultivars which the sequences were derived from and the number of transitions, transversions and indels detected is provided. To view the sequence assembly, click on 'contig list' and select 'Contig1'. Alternatively, the complete results can be downloaded to a local computer as a compressed file.

The main results page is divided into two frames, which can be expanded by dragging the central bar, and lists three types of information (**Figs. 15.4** and **15.5**). The right frame provides the complete sequence assembly with sequences running vertically down the page. The base position of the assembly is to the left of this frame and predicted SNPs are highlighted in colour with the SNP redundancy score and the SNP type towards the right of the frame. The left frame provides a list of the assembled sequences coded alphabetically and by case. The plant variety or cultivar

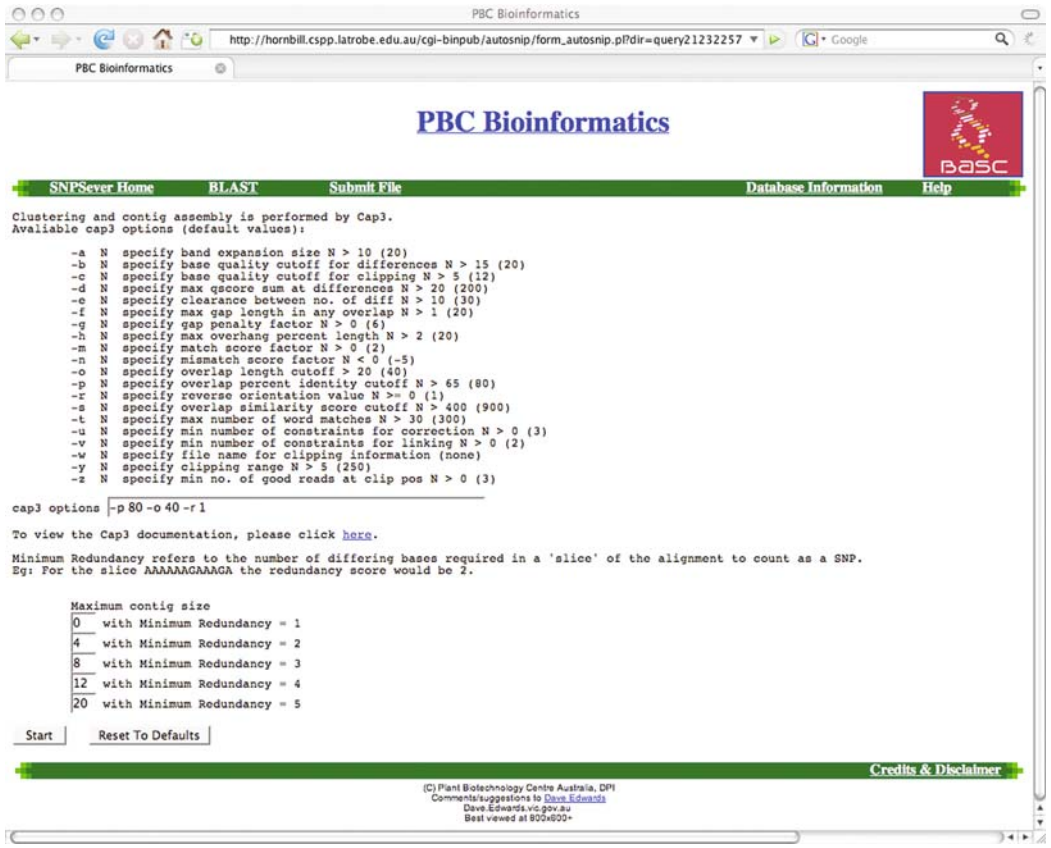


Fig. 15.2. The SNPServer sequence assembly and SNP discovery parameter page. CAP3 options for sequence assembly and minimum SNP redundancy scores may be set here.

names are appended to the name to ease interpretation of the results. Each name is linked to the original sequence maintained at NCBI. Within the left frame, below the list of sequence names there is a summary of the SNPs identified within the assembly. Each highlighted SNP includes the respective base pair position in the assembly to the left and the alphabetic code for each sequence at the top. The co-segregation and weighted co-segregation scores for each of the SNPs are also shown (**Fig. 15.5**).

Within this example, three clear haplotypes can be observed. The three haplotypes formed may represent the A, B and D genomes of wheat, or may be multiple genes from a gene family. The SNP at base pair 292 appears to be triallelic; however, this is unlikely and as it represents the last nucleotide in sequence A, it is probably due to sequence error. The SNP at base 332 has a low co-segregation score due to the N in sequence b at this position. The variety Chinese Spring is represented by more than one haplotype, suggesting that the polymorphism may be between duplicate or homeologous loci.

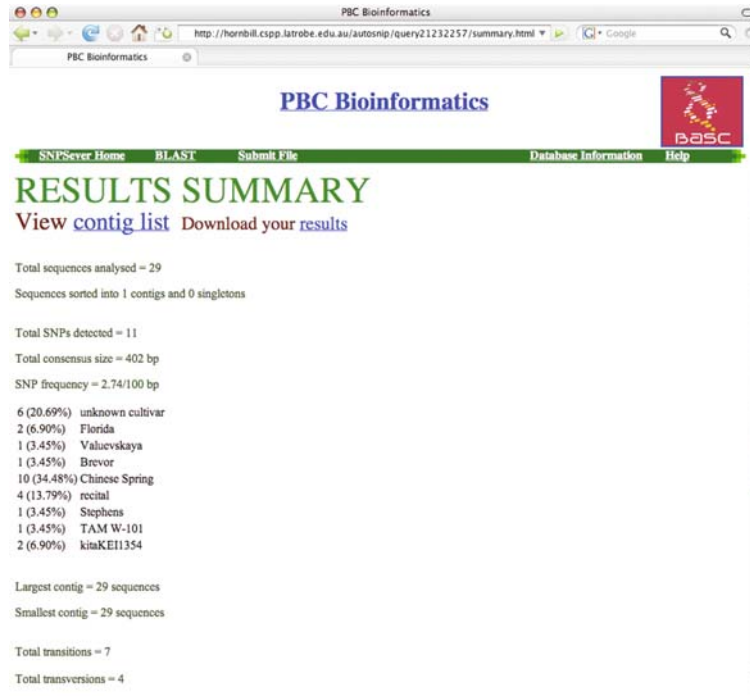


Fig. 15.3. Summary results from a SNPServer search.

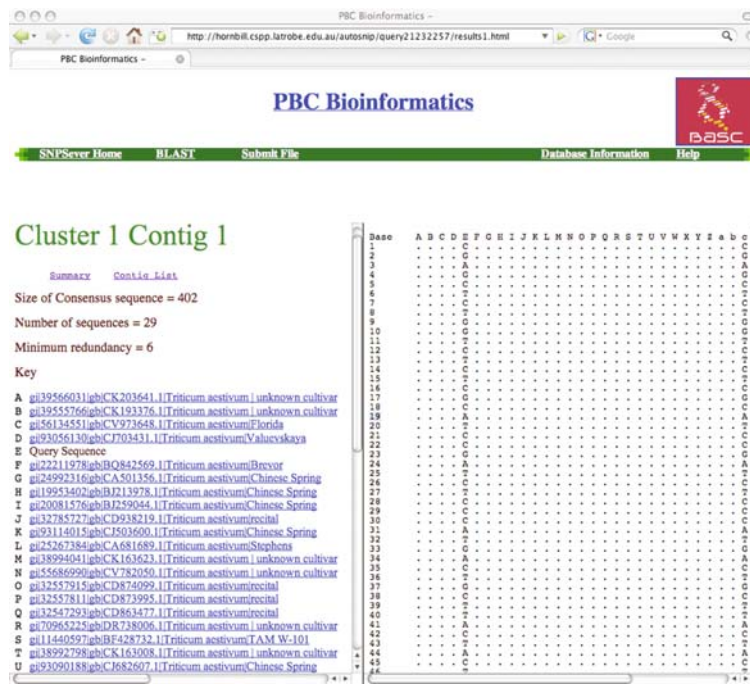


Fig. 15.4. Example SNPServer results (I). The *left* frame displays the details of the assembled sequences. The *right* frame shows the assembled sequences running vertically with base No. 1 at the *top*.

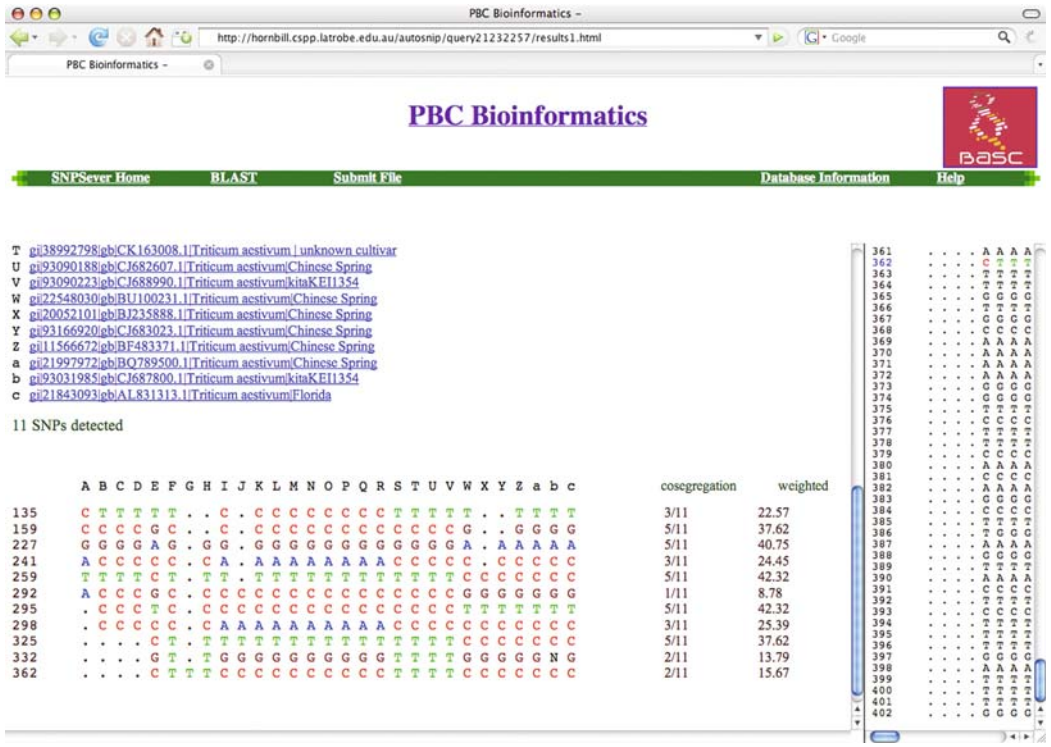


Fig. 15.5. Example SNPServer results (II). The *lower* part of the *left* frame displays the summary of identified SNPs, with co-segregation and weighted co-segregation scores.

3.2. dbSNP: Identification of a Previously Validated SNP Associated with Human Disease

Open the dbSNP web site (http://www.ncbi.nlm.nih.gov/SNP/index.html).There are a variety of methods to restrict or refine searches of dbSNP. **Table 15.1** lists some examples and available search fields. Enter rs334 into the search box and click on 'Search'. This is the SNP id for the polymorphism, which causes sickle cell mutation in the HBB (haemoglobin beta gene). This search should return complete results for this polymorphism (**Figs. 15.6**, **15.7**, **15.8** and **15.9**) (*see* **Note 6**).

The main results page is organised into several sections providing details of the SNPs. Sections include introductory summary information, submission information, FASTA sequence, GeneView, links to related Maps, links to additional NCBI information, allele frequency information for different population groups and a validation summary. Further links provide further details for each section. The submission section (**Fig. 15.6**) provides information on the submission with the greatest amount of flanking sequence, assay and submitter identifications including links to these, the validation status, the orientation of the sequence, the specific alleles detected and the 30-bp flanking sequence 5′ and 3′ up and downstream of the SNP. Information on the submission date, any

Table 15.1
Example dbSNP search fields

Example	Description
BRC* [gene name]	Search SNPs on all genes with names starting with the letter 'BRC' (i.e. BRCA1 and BRCA2)
1:5 [HET]	Search SNPs with heterozygosity between 1 and 5%
Coding non-synonymous [FUNC] AND 1 [CHR]	Search SNPs with function class 'coding non-synonymous' located on chromosome 1
1 [CHR] OR 2 [CHR]	Search all SNPs on chromosome 1 or 2
1 [CHR] OR 2 [CHR] NOT unknown [METHOD]	Search all SNPs on chromosome 1 or 2 detected by all methods except 'unknown'.
1 [WEIGHT] AND (1 [CHR] OR 2 [CHR]) NOT (unknown [METHOD] OR computed [METHOD])	Search all SNPs with weight 1 on chromosome 1 or 2 detected by all methods except 'unknown' or 'computed'.

dates where the submission was updated, build number of the original submission and molecule type are also provided in this section. The middle part of the results page provides information on the FASTA sequence, gene view and integrated maps (**Fig. 15.7**). The FASTA section provides the sequence submitted in rs334 in FASTA format. It provides the location of the SNP and

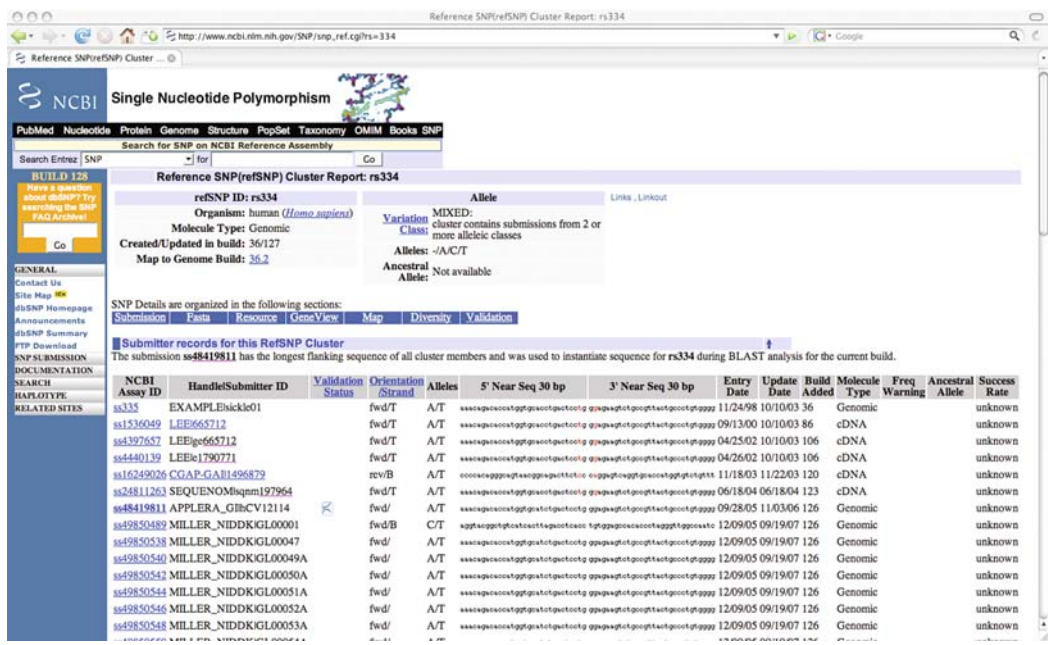


Fig. 15.6. Summary results and submitter information for SNP rs344.

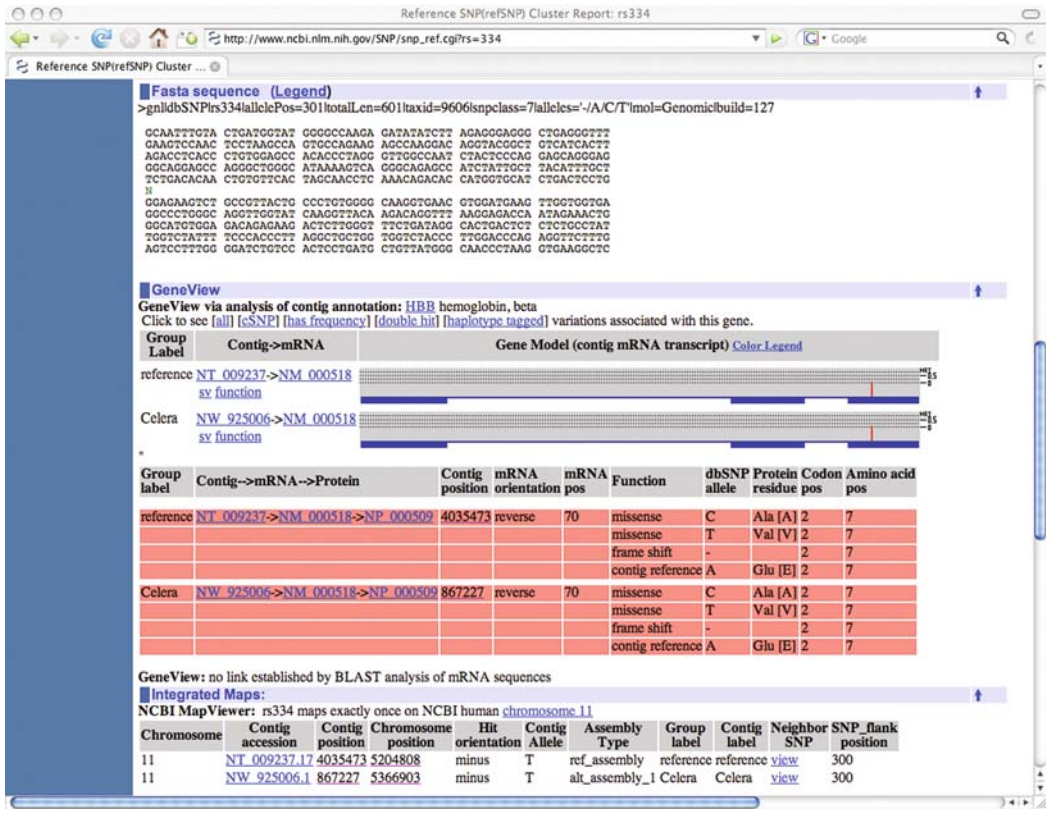


Fig. 15.7. FASTA sequence, GeneView and Integrated Maps for SNP rs344.

the available flanking sequence. This information could be used to design primers for a detection assay for this polymorphism. The GeneView section provides a gene model and the position of the SNP within the sequence. It also provides information on the effect of the SNP on the coding sequence, i.e. whether it is synonymous or non-synonymous and therefore whether it will cause an amino acid substitution. There are links from this section to the full gene sequence and other SNPs identified within this sequence. Integrated maps provide the location of the SNP with the link to the genome view and information on the mapped contigs and the neighbouring SNPs. The lower part of the results page provides information on the validation, links to NCBI resources and population diversity (**Fig. 15.8**). NCBI Resources link provides links to other NCBI web-pages, including links to submitter information and BLAST hits for the sequence. Population diversity gives information on the genotype frequency of the different SNP alleles in populations from the different continents. The validation summary provides information on any validation information for the SNP.

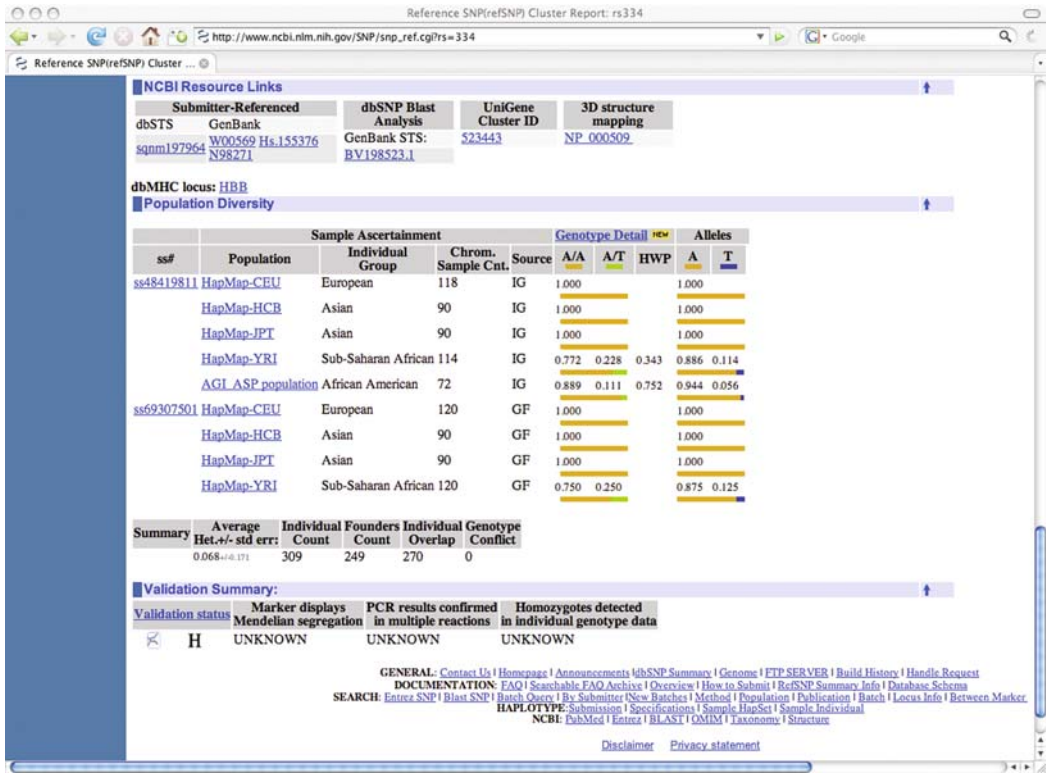


Fig. 15.8. NCBI resource links, population diversity and validation summary for SNP rs344.

3.3. Identification of PCR Amplification Primers for Bovidae SSRs Using the SSR Taxonomy Tree Tool

Open the SSR taxonomy tree web site (http://bioinformatics.pbcbasc.latrobe.edu.au/cgi-bin/ssr_taxonomy_browser.cgi) (*see* **Note 4**). Within the search box, type 'bovidae' and click on Search. This should return one match. Click on Bovidae to view the SSRs present. All identified SSR primer pairs within the Bovidae subtaxa can be downloaded by clicking the Download button (*see* **Note 7**).

To identify SSR primers for the sub taxa *Ovis* (sheep), browse the taxonomic tree by clicking on Caprinae, then *Ovis* (*see* **Note 8**). There are 3,081 SSR primer pairs available for five *Ovis* species, the majority of which are from *O. aries*, the domestic sheep. Click on 'view' to list the SSRs (**Fig. 15.9**). Information is provided on the GenBank accession number, with a link to the original sequence, details of the SSR repeat type, motif, length and position in the sequence. Information on the SSR amplification primers include the primer sequence, relative position in the original sequence, expected product size, GC content, Tm and complementarity score. By clicking on the column heading, the user can sort the data according to the parameter they are most interested in, such as SSR length or type. The information may be downloaded by clicking on 'Download'. This produces a text file containing the SSR and primer pair details, which may be saved and opened in notepad or as a tab delimited file in spreadsheet software.

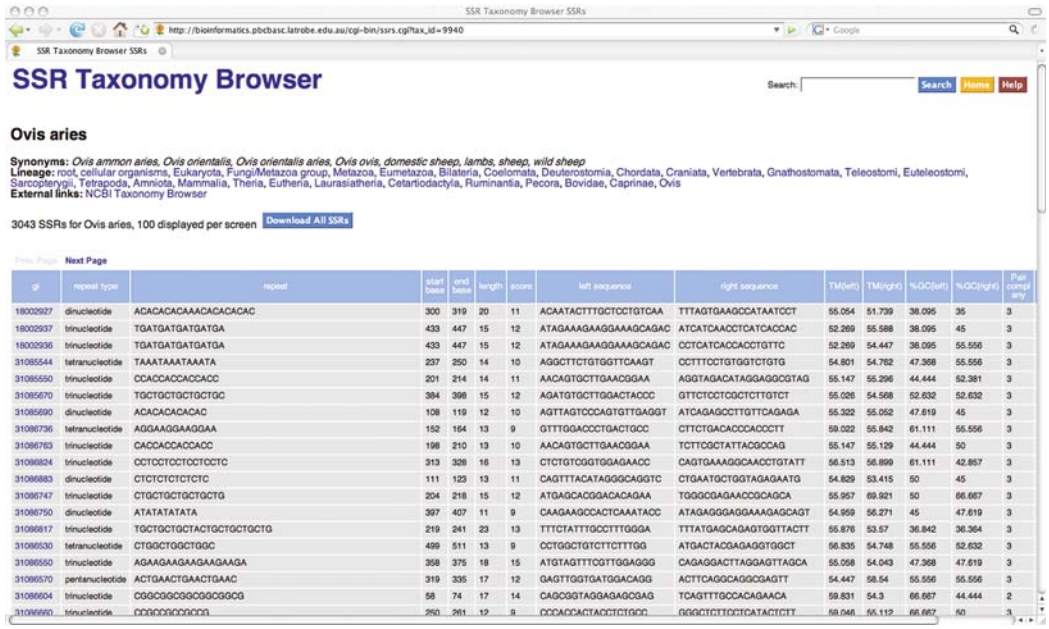


Fig. 15.9. List of SSR primer pair results for *Ovis aries*.

4. Notes

1. SSRs are also known as microsatellites, following the method of their initial identification. They are now more commonly referred to as SSRs.
2. PolyPhred integrates Phred base calling and peak information with Phrap *(38)* generated sequence alignments *(16, 17)*. Alignments are viewed using Consed *(39)*. More recently, this approach has been extended to include Bayesian statistical analysis. PolyBayes is a fully probabilistic SNP detection algorithm that calculates the probability that discrepancies at a given location of a multiple alignment represent true sequence variations as opposed to sequencing errors. The calculation takes into account the alignment depth, the base calls in each of the sequences, the associated base quality values, the base composition in the region, and the expected a priori polymorphism rate.
3. The optimal parameters for SNP discovery are dependent on the available sequence data and the final application of the SNPs. Where submitted sequence quality is known to be near perfect, the redundancy score may be reduced to 1 to display

all polymorphic sites, even if they occur only once in an alignment. Where the object is to assess the genetic diversity of a gene between distantly related species, less stringent CAP3 sequence assembly parameters may be applied. It is recommended that users first apply the default SNPServer parameters and return to modify the BLAST, CAP3 and SNP discovery parameters and compare the results.

4. Alternative versions of the software are available from http://acpfg.imb.uq.edu.au/
5. The default option is to assemble all the sequences identified using BLAST. Alternatively, individual sequences may be selected for assembly by clicking the boxes to the left of the sequence name, then click on 'Run SNP detection on selected sequences'.
6. Alternatively, the information on this SNP could have been identified by performing a BLAST search of dbSNP with the HBB gene sequence.
7. SSRs may be viewed and downloaded. When datasets are smaller than 3 Mb (~21,000 SSRs) they may be viewed or downloaded. When datasets are smaller than 5 Mb (~34,000 SSRs), they may be downloaded only. Some larger sets have been compressed and may be downloaded from the Large Downloads page. If you require a set larger than 5 Mb to be included in Large Downloads, which is not present already, please contact Dave Edwards (Dave.Edwards@acpfg.com.au).
8. Alternatively the search can be repeated using the term *Ovis* or sheep.

References

1. Brumfield, R. T., Beerli, P., Nickerson, D. A., and Edwards, S. V. (2003) The utility of single nucleotide polymorphisms in inferences of population history. *Trends Ecol Evol* **18,** 249–56.
2. Collins, A., Lau, W., and De la Vega, F. M. (2004) Mapping genes for common diseases: The case for genetic (LD) maps. *Hum Hered* **58**, 2–9.
3. Rafalski, A. (2002) Applications of single nucleotide polymorphisms in crop genetics. *Curr Opin Plant Biol* **5,** 94–100.
4. Gupta, P. K., Roy, J. K., and Prasad, M. (2001) Single nucleotide polymorphisms: a new paradigm for molecular marker technology and DNA polymorphism detection with emphasis on their use in plants. *Curr Sci* **80,** 524–35.
5. Clifford, R., Edmonson, M., Hu, Y., Nguyen, C., Scherpbier, T., and Buetow, K. H. (2000) Expression-based genetic/physical maps of single nucleotide polymorphisms identified by the cancer genome anatomy project. *Genome Res* **10,** 1259–65.
6. Deutsch, S., Iseli, C., Bucher, P., Antonarakis, S. E., and Scott H. S. (2001) A cSNP map and database for human chromosome 21. *Genome Res* **11,** 300–7.
7. Edwards, D., Forster, J. W., Chagné, D., and Batley, J. (2007) What are SNPs? in *Association Mapping in Plants* (Oraguzie N. C., Rikkerink E. H. A, Gardiner S. E. and De Silva H. N., Eds.), Springer New York, pp 41–52.

8. Syvanen, A. C. (2001) Genotyping single nucleotide polymorphisms. *Nat Rev Genet* **2,** 930–42.
9. The International HapMap Consortium. (2003) The international HapMap project. *Nature* **426,** 789–96.
10. Buetow, K. H., Edmonson, M. N., and Cassidy, A. B. (1999) Reliable identification of large numbers of candidate SNPs from public EST data. *Nature Genet* **21,** 323–5.
11. Gu, Z., Hillier, L., and Kwok, P.-Y. (1998) Single nucleotide polymorphism hunting in cyberspace. *Hum Mutat* **12,** 221–5.
12. Picoult-Newberg, L., Ideker, T. E., Pohl, M. G., Taylor, S. L., Donaldson, M. A., Nickerson, D. A., and Boyce-Jacino M. (1999) Mining SNPs from EST databases. *Genome Res* **9,** 167–74.
13. Taillon-Miller, P., Gu, Z., Li, Q., Hillier, L., and Kwok, P.-Y. (1998) Overlapping genomic sequences: a treasure trove of single-nucleotide polymorphisms. *Genome Res* **8,** 748–754.
14. Adams, M. D., Kerlavage, A. R., Fleischmann, R. D., Fuldner, R. A., Bult, C. J., Lee, N. H., Kirkness, E. F., Weinstock, K. G., Gocayne, J. D., White, O., Sutton, G., Blake, J. A., Brandon, R. G., Chiu, M. W., Clayton, R. A., Cline, R. T., Cotton, M. D., Earlehughes, J., Fine, L. D., Fitzgerald, L. M., Fitzhugh, W. M., Fritchman, J. L., Geoghagen, N. S. M., Glodek, A., Gnehm, C. L., Hanna, M. C., Hedblom, E., Hinkle, P. S., Kelley, J. M., Klimek, K. M., Kelley, J. C., Liu, L. I., Marmaros, S. M., Merrick, J. M., Morenopalanques, R. F., Mcdonald, L. A., Nguyen, D. T., Pellegrino, S. M., Phillips, C. A., Ryder, S. E., Scott, J. L., Saudek, D. M., Shirley, R., Small, K. V., Spriggs, T. A., Utterback, T. R., Weldman, J. F., Li, Y., Barthlow, R., Bednarik, D. P., Cao, L. A., Cepeda, M. A., Coleman, T. A., Collins, E. J., Dimke, D., Feng, P., Ferrie, A., Fischer, C., Hastings, G. A., He, W. W., Hu, J. S., Huddleston, K. A., Greene, J. M., Gruber, J., Hudson, P., Kim, A., Kozak, D. L., Kunsch, C., Ji, H. J., Li, H. D., Meissner, P. S., Olsen, H., Raymond, L., Wei, Y. F., Wing, J., Xu, C., Yu, G. L., Ruben, S. M., Dillon, P. J., Fannon, M. R., Rosen, C. A., Haseltine, W. A., Fields, C., Fraser, C. M., and Venter, J. C. (1995) Initial assessment of human gene diversity and expression patterns based upon 83-million nucleotides of cDNA sequence. *Nature* **377,** (Suppl.) 3–17.
15. Dawson, E., Chen, Y., Hunt, S., Smink, L. J., Hunt, A., Rice, K., Livingston, S., Bumpstead, S., Bruskiewich, R., Sham, P., Ganske, R., Adams, M., Kawasaki, K., Shimizu, N., Minoshima, S., Roe, B., Bentley, D., and Dunham, I. (2001) A SNP resource for human chromosome 22: extracting dense clusters of SNPs from the genomic sequence. *Genome Res* **11,** 170–8.
16. Ewing, B., and Green, P. (1998) Base-calling of automated sequencer traces using phred. II. Error probabilities. *Genome Res* **8,** 186–94.
17. Ewing, B., Hillier, L., Wendl, M. C., and Green, P. (1998) Base-calling of automated sequencer traces using phred. I. Accuracy assessment. *Genome Res* **8,** 175–85.
18. Batley, J., Barker, G., O'Sullivan, H., Edwards, K. J., and Edwards, D. (2003) Mining for single nucleotide polymorphisms and insertions/deletions in maize expressed sequence tag data. *Plant Physiol* **132,** 84–91.
19. Barker, G., Batley, J., O'Sullivan, H., Edwards, K. J., and Edwards, D. (2003) Redundancy based detection of sequence polymorphisms in expressed sequence tag data using autoSNP. *Bioinformatics* **19,** 421–2.
20. Savage, D., Batley, J., Erwin, T., Logan, E., Love, C. G., Lim, G. A. C., Mongin, E., Barker, G., Spangenberg, G.C., and Edwards, D. (2005) SNPServer: a real-time SNP discovery tool. *Nucleic Acids Res* **33,** W493–5.
21. Huang, X. and Madan, A. (1999) CAP3: a DNA sequence assembly program. *Genome Res* **9,** 868–77.
22. Altschul, S. F., Gish, W., Miller, W., Myers, E. W., and Lipman, D. J. (1990) Basic local alignment search tool. *J Mol Biol* **215,** 403–10.
23. Sherry, S. T., Ward, M., and Sirotkin, K. (1999) dbSNP-Database for single nucleotide polymorphisms and other classes of minor genetic variation. *Genome Res* **9,** 677–9.
24. Smigielski, E. M., Sirotkin, K., Ward, M., and Sherry, S. T. (2000) dbSNP: a database of single nucleotide polymorphisms. *Nucleic Acids Res* **28,** 352–5.
25. Tóth, G., Gáspári, Z., and Jurka, J. (2000) Microsatellites in different eukaryotic genomes: survey and analysis. *Genome Res* **10,** 967–81.
26. Mortimer, J., Batley, J., Love, C., Logan, E., and Edwards, D. (2005) Simple sequence repeat (SSR) and GC distribution in the *Arabidopsis thaliana* genome. *J Plant Biotechnol* **7,** 17–25.

27. Katti, M. V., Ranjekar, P. K., and Gupta, V. S. (2001) Differential distribution of simple sequence repeats in eukaryotic genome sequences. *Mol Biol Evol* **18,** 1161–7.
28. Schlötterer, C. (2000) Evolutionary dynamics of microsatellite DNA. *Nucleic Acids Res* **20,** 211–5.
29. Tautz, D. (1989) Hypervariability of simple sequences as a general source for polymorphic DNA markers. *Nucleic Acids Res* **17,** 6463–71.
30. Powell, W., Machray, G. C. and Provan, J. (1996) Polymorphism revealed by simple sequence repeats. *Trends Plant Sci* **1,** 215–22.
31. Edwards, K. J., Barker, J. H. A., Daly, A., Jones, C., and Karp, A. (1996) Microsatellite libraries enriched for several microsatellite sequences in plants. *Biotechniques* **20,** 758–60.
32. Robinson, A. J., Love, C. G., Batley, J., Barker, G., and Edwards, D. (2004) Simple sequence repeat marker loci discovery using SSRPrimer. *Bioinfomatics* **20,** 1475–6.
33. Jewell, E., Robinson, A., Savage, D., Erwin, T., Love, C. G., Lim, G. A. C., Li, X., Batley, J., Spangenberg, G. C., and Edwards, D. (2006) SSR primer and SSR taxonomy tree: biome SSR discovery. *Nucleic Acids Res* **34,** W656–9.
34. Abajian, C. (1994) SPUTNIK, http://abajian.net/sputnik/.
35. Rozen, S., and Skaletsky, H. J. (2000) Primer3 on the WWW for general users and for biologist programmers, in: *Bioinformatics Methods and Protocols: Methods in Molecular Biology* (Krawetz S., and Misener S., Eds.), Humana Press, Totowa, NJ, pp 365–86.
36. Stajich, J. E., Block, D., Boulez, K., Brenner, S. E., Chervitz, S. A., Dagdigian, C., Fuellen, G., Gilbert, J. G. R., Korf, I., Lapp, H., Lehvaslaiho, H., Matsalla, C., Mungall, C. J., Osborne, B. I., Pocock, M. R., Schattner, P., Senger, M., Stein, L. D., Stupka, E., Wilkinson, M. D., and Birney, E. (2002) The Bioperl toolkit: Perl modules for the life sciences. *Genome Res* **12,** 1611–8.
37. Wilson, I. D., Barker, G. L. A., Beswick, R. W., Shepherd, S. K., Lu, C., Coghill, J. A., Edwards, D., Owen, P., Lyons, R., Parker, J. S., Lenton, J. R., Holdsworth, M. J., Shewry, P. R., and Edwards, K. J. (2004) A transcriptomics resource for wheat functional genomics. *Plant Biotechnol J***2,** 495–506.
38. Green, P. (1994) Phrap, unpublished. www.phrap.org.
39. Gordon, D., Abajian, C., and Green, P. (1998) Consed: a graphical tool for sequence finishing. *Genome Res***8,** 195–202.

Chapter 16

Analysis of Transposable Element Sequences Using CENSOR and RepeatMasker

Ahsan Huda and I. King Jordan

Abstract

Eukaryotic genomes are full of repetitive DNA, transposable elements (TEs) in particular, and accordingly there are a number of computational methods that can be used to identify TEs from genomic sequences. We present here a survey of two of the most readily available and widely used bioinformatics applications for the detection, characterization, and analysis of TE sequences in eukaryotic genomes: CENSOR and RepeatMasker. For each program, information on availability, input, output, and the algorithmic methods used is provided. Specific examples of the use of CENSOR and RepeatMasker are also described. CENSOR and RepeatMasker both rely on homology-based methods for the detection of TE sequences. There are several other classes of methods available for the analysis of repetitive DNA sequences including de novo methods that compare genomic sequences against themselves, class-specific methods that use structural characteristics of specific classes of elements to aid in their identification, and pipeline methods that combine aspects of some or all of the aforementioned methods. We briefly consider the strengths and weaknesses of these different classes of methods with an emphasis on their complementary utility for the analysis of repetitive DNA in eukaryotes.

Key words: Transposable elements, sequence analysis, bioinformatics, Repbase, CENSOR, RepeatMasker.

1. Introduction

Transposable elements (TE) are repetitive DNA sequences capable of moving from one chromosomal locus to another. The ubiquity of TEs has been appreciated for some time; they have been found in the genomes of a wide variety of species from all three domains of life. However, one of the major revelations of eukaryotic genome sequencing projects was the staggering abundance of

David Posada (ed.), *Bioinformatics for DNA Sequence Analysis,* Methods in Molecular Biology 537

DOI 10.1007/978-1-59745-251-9_16

TE-related sequences in large genomes. For instance, approximately one half of the human genome sequence was shown to consist of the remnants of TE insertion events *(1)*. In light of the sustained efforts underway to sequence and characterize numerous eukaryotic genomes, the prevalence of TEs necessitates the development and use of computational tools aimed at their detection, characterization, and analysis. After all, it is simply not possible to fully comprehend the structure, function, and evolution of eukaryotic genomes without a deep understanding of their TEs.

The most commonly used programs for the detection and analysis of TE sequences employ comparisons of genomic sequences to a library of consensus sequences that represent families of known repetitive (transposable) elements. This is the so-called homology-based method for the detection of TEs in genomic sequence. The Repbase Update *(2, 3)* is a comprehensive database of known eukaryotic repetitive sequence elements maintained by the Genetic Information Research Institute (GIRI; http://www.girinst.org). The developers of the Repbase Update, led by Jerzy Jurka, pioneered computational approaches toward the automatic detection of TEs in genomic sequences. DNA sequence searches against very early versions of Repbase, aimed primarily at the detection of Alu elements, were first carried out by the Pythia server *(4, 5)*. The Pythia server later gave way to the program CENSOR *(6, 7)*, which is still maintained and distributed by the GIRI. The tight integration of CENSOR with the Repbase Update library provides the user with access to the latest available TE annotations, which are constantly being updated at the GIRI. In addition to identifying known TEs in genomic sequence, CENSOR also provides for the de novo identification of simple sequence repeats that are characteristic of low complexity DNA regions *(8)*.

Arian Smit's RepeatMasker is another widely used program that identifies the location and identity of TEs in genomic sequence via searches against the Repbase Update library *(9)*. RepeatMasker employs a similar approach to compare genomic sequences against Repbase as the CENSOR program does. Additionally, RepeatMasker incorporates a great deal of *ad hoc* post-processing in order to try and ensure the best representation of TEs as single contiguous regions in genomic sequence. RepeatMasker has been used to annotate the TEs of numerous eukaryotic genomes, including the human genome sequence, and static releases of RepeatMakser annotations are widely distributed on various genome databases. Insight gained from RepeatMasker analyses has been critical to the field of genomics.

In this chapter, we will provide specific information on, and examples of, the use of the programs CENSOR and RepeatMasker along with a description of several other complementary classes of methods available for the analysis of repetitive DNA sequences.

1.1. Complementary Methods

CENSOR and RepeatMasker represent one general class of methods for the detection and analysis of repetitive DNA sequences. There are several additional classes of methods for the analysis of repetitive DNA: (i) de novo methods, (ii) class-specific methods, and (iii) pipeline methods. All of the different classes of methods have different strengths and weaknesses with respect to their ability to detect and characterize TEs in eukaryotic genome sequences. As such, they may be considered to be complementary, and indeed when different methods are compared on the same query sequence, they are often found to identify substantially non-overlapping parts of the sequence as being repetitive. Thus, investigators should be careful not to rely overly on one method or another. Homology-based methods in particular are limited by the extent of knowledge that already exists concerning the repetitive elements of a given genome or evolutionary lineage. In other words, the TEs, or their relatives, must have been previously characterized in order to be detected by homology-based methods. For this reason, these methods will perform poorly when applied to genomes that have many uncharacterized TE families. Homology-based methods will also be unable to detect novel TE families with distinct sequences. De novo methods, on the other hand, are ideal for identifying previously unknown repetitive DNA elements. However, de novo methods provide no information on the identity of these elements, or whether they are even TEs at all, and as such can be best used to simply mask repetitive elements. Clearly, homology-based methods are far better suited for investigations into the biology and genome dynamics of the TEs themselves.

1.1.1. De Novo Methods

Another general class of applications for identifying repeats in genomic sequence entails the so-called de novo methods that identify repeats by comparing genomic query sequences against themselves. Repeats are characterized in this way by clustering the similar groups of sequences that emerge from self comparison. De novo methods are interesting from an historical perspective because they represent the computational analogs of the re-association kinetic experiments that were first used to demonstrate the repetitive nature of eukaryotic genomes *(10)*.

De novo methods are naïve in the sense that they do not require any prior knowledge of the repetitive elements that may be present in the query sequence. This has the effect of eliminating ascertainment biases leading to false negatives for unknown repetitive elements. So in the formal sense de novo methods represent the most sensitive approach for the detection of repetitive DNA, and the recently developed WindowsMasker de novo method *(11)* has the added advantage of being much faster than homology-based methods. However, to work properly de novo methods require long and complete (or nearly so) query

sequences (i.e., whole contigs or genomes). More importantly, these methods do not provide any information on the characteristics of the repeats that are detected. De novo methods will report repeats of very different classes, such as tandem repeats, large segmental duplications, and interspersed repeats (TEs), together without discriminating among them. In other words, de novo methods work well for the detection and/or masking of repeat elements but do not aid in their characterization or analysis. De novo methods are also generally ineffective in identifying repetitive elements that are in low copy number as well as relatively ancient repetitive elements that may be too divergent from one another to be recognized as repetitive. RECON is another de novo method available for the detection of repetitive DNA sequences *(12)*.

1.1.2. Class-Specific Methods

Class-specific methods are a relatively recent development in the detection and analysis of TE sequences. For these methods, experts in the analysis of TEs have taken advantage of particular genomic features characteristic of specific classes of elements to aid in their identification. This approach has been most widely implemented with the LTR_STRUC program that identifies members of the long terminal repeat (LTR) containing class of TEs by virtue of the direct repeat sequences that are present at both ends of the elements *(13, 14)*. A recent publication presents a newly implemented method for the identification of LTR elements in eukaryotic genomes based on the same underlying rationale as LTR_STRUC *(15)*. However, in addition to identifying full length elements, this new program can also identify solo LTRs.

Since these kinds of methods do not rely on sequence identity (similarity) searches, they are particularly well suited to the identification of novel element families and low copy number elements. However, these methods are limited to families of elements that possess well-defined structural characteristics such as LTR elements and miniature-inverted repeat containing TEs (MITEs). Class-specific methods also enable the detection of novel TE sequences from a given element class while allowing for a deep interrogation of elements from that class. On the other hand, these methods will be particularly sensitive to sequence changes that accumulate after TE insertion and obscure the structural characteristics, such as inverted repeats, that they use to identify TEs.

1.1.3. Pipeline Methods

Pipeline methods, which combine aspects of all the aforementioned approaches to TE detection, probably represent the most rigorous and accurate class of method available for the annotation of TE sequences in eukaryotic genomes. Examples of pipeline methods are the MITE analysis toolkit (MAK) *(16)* and a more

recently proposed pipeline method, which promises to provide the most accurate and reliable annotations of TE sequences in eukaryotic genomes to date *(17)*. While these methods are very powerful in principle, they are also among the least accessible to the user because their use entails far more effort than any of the other single methods. Because pipeline methods integrate so many distinct applications, they also require a high level of sustained development and maintenance. Pipeline methods may well become the standard approach for genome annotation and serve the community best by providing static TE annotations of eukaryotic genomes as opposed to readily usable tools for investigators to query their own sequences of interest.

2. Program Usage

2.1. CENSOR

CENSOR allows for the identification and characterization of repetitive elements in genomic sequences. CENSOR can be used to mask repetitive sequences to allow for the more efficient use of downstream applications that are confounded by the presence of repeats and it can also be used to identify and characterize repetitive sequences in order to study the biology of the elements themselves.

2.1.1. Purpose

2.1.2. Availability

CENSOR is freely available to download from the GIRI for local installation (http://www.girinst.org/censor/download.php). CENSOR can be run locally using Unix type computer operating systems. Running CENSOR locally requires the installation of a local version of Repbase, which is optionally included in the download package, as well as the WU-BLAST package *(18)*. CENSOR can also be run from a server on the GIRI website (http://www.girinst.org/censor/index.php).

2.1.3. Input

Sequences in FASTA, GENBANK, and EMBL formats can be submitted to CENSOR by uploading a file to their server or by pasting them in the query textbox. CENSOR accepts DNA as well as protein sequences as input and decides the version of BLAST to use given a particular query sequence. One or more sequences can be submitted in a particular query.

2.1.4. Output

CENSOR runs yield a number of distinct kinds of output including (a) a repeat map indicating the location of repeats on the query sequence, (b) annotation of the repeat location, type, and its similarity and positive score values, and (c) a "masked" sequence file that returns the repetitive sequences replaced by Ns or Xs.

2.1.5. Method

CENSOR uses WU-BLAST *(18)* or NCBI BLAST *(19)* algorithms to search the query sequence against the Repbase Update library of repetitive sequences. CENSOR can be run on three different speed/sensitivity settings (*see* **Note 1**). It can automatically run an appropriate version of BLAST such as BLASTN, BLASTP, BLASTX, and TBLASTN in order to accommodate the various input types used for querying repetitive elements. This feature adds flexibility to the algorithm in contrast to RepeatMasker, which only uses DNA sequences in its searches. All options available through BLAST can also be incorporated in CENSOR searches. CENSOR uses an information theoretic method to detect simple sequence repeats such as satellite DNA and low complexity sequences. CENSOR also post-processes data to give an interactive positional map of the query sequence (similar to the NCBI BLAST web interface). In addition, it calculates the similarity values and positive score values for alignments between query and element consensus sequences. The similarity value can be used to approximate the evolutionary of the age of the TEs.

2.2. RepeatMasker

RepeatMasker serves to identify, characterize, and mask repetitive elements in genomic sequences. It is most often used to simply mask identified repeats in genomic sequence so that other analyses can be run on the resulting non-repetitive DNA sequences. However, RepeatMasker also characterizes repeats by class, family, and individual element name based on the Repbase library, and this information is critical to the study of TEs. Divergence values between TEs and their family consensus sequences are also provided and these can be used to determine the relative age of the elements.

2.2.1. Purpose

2.2.2. Availability

RepeatMasker can be run in two different ways. The program can be downloaded from http://www.repeatmasker.org/RMDownload.html and installed locally, or it can be run on a web server http://www.repeatmasker.org/cgi-bin/WEBRepeatMasker. To install and run RepeatMasker locally, users will also need to install a local copy of the Repbase library as well as the programs WU-BLAST *(18)* and CROSS_MATCH *(20)*.

2.2.3. Input

RepeatMasker works only on DNA sequences and the query sequences have to be in FASTA format. Sequences can be submitted using a file with one or more sequences or by pasting the sequence(s) in the submission box. Extremely long sequences, or files with numerous sequences, will be automatically broken down into batches to be run by RepeatMakser.

2.2.4. Output

RepeatMasker runs yield three files: (a) annotation of the location, type, and percent divergence of repeat from the consensus sequence, (b) a sequence file that has the repetitive sequences replaced by Ns or

Xs, and (c) a summary of the repetitive content of the query sequence. Additional output files, including alignments between query and consensus sequences, can be optionally included.

2.2.5. Method

RepeatMasker scans the query sequence using the program CROSS_MATCH *(20)* against the library of consensus sequences provided by Repbase Update. CROSS_MATCH implements the Smith-Waterman (SW) dynamic programming algorithm *(21)* that guarantees optimal pairwise sequence alignments. Using CROSS_MATCH, a score matrix is first constructed based on exact word matches between the library sequences and the query sequence. This is then expanded to include a "band" of sequences that surround the exact match. The band is based on the overlap of SW scoring matrices. The width of the band, and thus the sensitivity of RepeatMasker, can be adjusted using different speed settings to allow for wider or narrower acceptance of sequences surrounding the band. Since there can be many consensus sequences in the Repbase Update library that match the same region of the query sequence, the search engines return the matrices that have less than 80–90% overlap with each other. Typically the sequence with the highest SW score is selected for annotation after various approximation improvements. RepeatMasker can also use WU-BLAST to search against Repbase to improve the speed of searches *(22)*. Simple repeats are detected by computing the AT or GC content for overlapping windows of 200 bp and then checking for characteristics attributed to most simple repeats. RepeatMasker uses stringent criteria for identifying simple repeats and low-complexity DNA, which can result in omission of some repeats.

3. Examples

3.1. CENSOR

We provide an example of running CENSOR from the GIRI web server. The URL http://www.girinst.org/censor/index.php points to the CENSOR submission page (**Fig. 16.1**). We used a 2-kb DNA sequence from the proximal promoter region of the human hydroxysteroid (17-beta) dehydrogenase 13 gene as an example query (Genbank mRNA accession NM_178135). The FASTA format sequence is pasted into the submission page textbox as shown; note that a file with the sequence could also be uploaded using the Browse and Submit buttons shown (**Fig. 16.1**). For the purposes of this search, the "Mammalia" option of the "Sequence source" is chosen. This option specifies which subset of Repbase will be searched and in this case the subset will include all repeat sequences that are common to mammals as

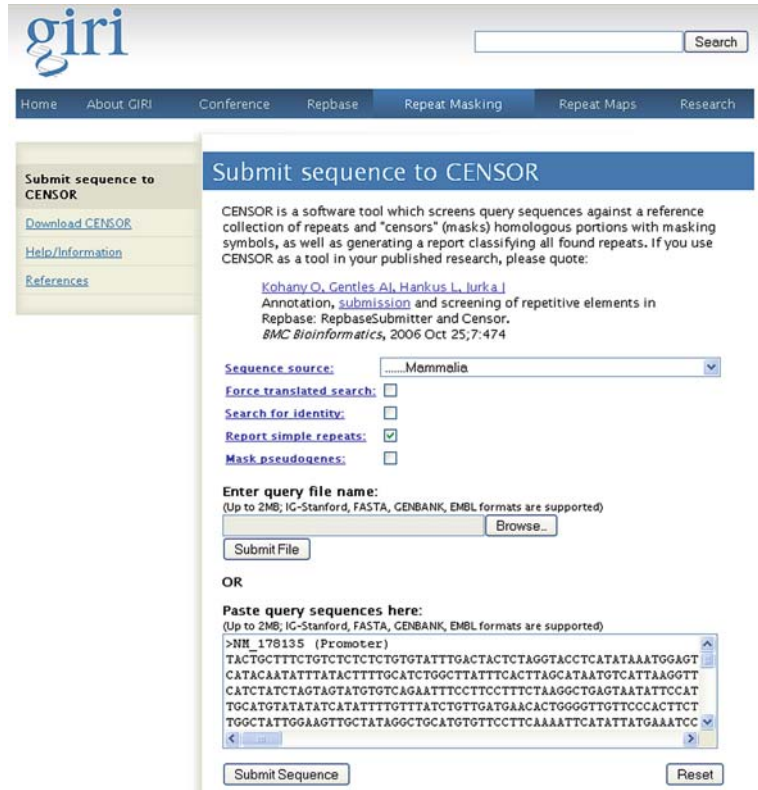


Fig. 16.1. CENSOR web server query submission page.

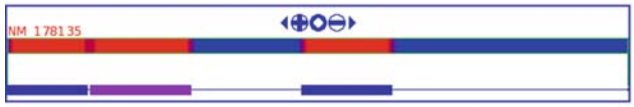
well as those specific to individual mammalian species. The "Report simple repeats" option is also selected to identify simple sequence repeats. Since the sequence is non-coding a translated search is not used. Neither the option "Search for identity," which forces the program to search for only identical or nearly identical sequences, nor the option "Mask pseudogenes," which searches for pseudogenes, is selected in this example.

Once the query sequence is pasted (or uploaded) and the appropriate options are selected, the search is run using the "Submit Sequence" button (**Fig. 16.1**). There are several output displays provided by CENSOR. CENSOR post-processes data to give an interactive positional map of the query sequence along with a summary table of identified elements (**Fig. 16.2**). On the positional map, the query sequence is represented by the horizontal bar with red representing repetitive (masked) DNA and blue representing non-repetitive DNA. The individual repeats and their annotations are shown below the bar; mouse-overs yield the element name and classification, and clicking on the element links to its Repbase entry. A masked version of the sequence is also provided (**Fig. 16.3**), as are alignments of the query sequence with

Map of Hits

SVG viewer is required to view graphical representation of the map as Scalable Vector Graphics (SVG plot).

NM_178135 (SVG Plot; Alignments; Masked)



Name	From	To	Name	From	To	Class	Dir	Sim	Pos	Score
NM_178135	3	260	L1MB4	448	714	Non-LTR/L1	c	0.8605	0.86	1597
NM_178135	269	592	MLT1A0	19	374	ERV/ERV3	d	0.7798	0.78	1369
NM_178135	948	1239	AluSq	1	282	Non-LTR/SINE/SINE1	c	0.7746	0.77	1317

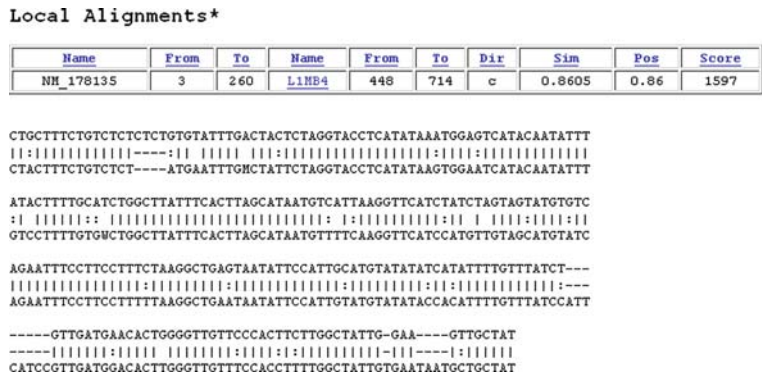
Fig. 16.2. CENSOR output repeat map and summary table. The location of repeats and their identity are shown as a graphical schematic and also listed line-by-line. Links to the Repbase entries for individual elements are provided in the schematic Figure and the table.

Masked Sequence

```
>NM_178135
TAXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXAGGCTGCAXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXGAAGCCACCCCACTTTAAAGTTGAGATTCCCGACTCCT
GGTCTAATGCTCTTTCCAGTATTATCACAAAGGGAACTGATGGTTCTGTATCCTAAAATCTCCCCCAAAC
TCAAGAGATTTTCAGGGAAATGGTCATCATGTGTAAAATAATTAGCCAGTTGAAATATTGATGCTGAGCT
TTTGTGAATAAATGAATCAAATAGCTAAGCTGGGTTCATTCAGACTTTATCTAAGTTTCTTTAAGCTTAT
CTCTACATTCTCCTTTTTCATATTGAAAAAAGGTAACAATTTCCAGAAAGATACCAACAGATCATTAGCT
GATGCAGTATCTCCAGAATTCTTTTTCTTTTTTTTTTXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXCCTAGAATTGTTTCTAGAGGT
GAAACTTCAAGGTGAAATATAGTACATAACTGCTTTTCAGATAAACAAGTCCAGAGAGCACACTCTCTTG
TGCTCTTGGCATCACTTGGCATCACTTCATATTTGAGGTGTTTCAAACCCATTAGAACACGTGAACAAGG
CCTGCTTCCAAAGCTGGCTTCCATCTGGTAGTCCCATTAACAACTGGGCACACCCCTTCCCTAGAGCTCT
GTGTAGACAGTACCTCCTCCCTAGGACTACACAAGGACTGAACCAGAAGGAAGAGGACAGAGCAAAGCCA
TGAACATCATCCTAGAAATCCTTCTGCTTCTGATCACCATCATCTACTCCTACTTGGAGTCGTTGGTGAA
GTTTTTCATTCCTCAGAGGAGAAAATCTGTGGCTGGGGAGATTGTTCTCATTACTGGAGCTGGGCATGGA
ATAGGCAGGCAGACTACTTATGAATTTGCAAAACGACAGAGCATATTGGTTCTGTGGGATATTAATAAGG
TAATGTATACATCTTCCAACTTTTTAAAGTCACAGAGTAAGATATGTATTTTAAGAATTATTTGACTTAC
CATCTACTTATCTTTGTATTTTTGTTTTTCAAAGTTTGATAAATTCCCTGGTCCCTTAGTCTGTATATGT
GTCAGGTTAGTTAGATGAAGGGAATGTAATTAAGAACTAAGCAGCGATTTTTATGACATGGTGTGCAGGT
TGATAGAAAGACTCAGGAGCCAGTCTCCTTCCAAGCTGCT
```

Fig. 16.3. Query sequence with repetitive elements masked by “X”s.

each repeat consensus sequence found in Repbase (**Fig. 16.4**). CENSOR also shows the Repbase records for all the repeats found in the query. Along with the alignments, CENSOR calculates the similarity values and positive score values (**Figs. 16.2** and **16.4**). The similarity value (Sim) reports the number of matches in the alignment normalized by the length of the alignment, while the positive value (Pos) reports the normalized number of alignment


Local Alignments*

Name	From	To	Name	From	To	Dir	Sim	Pos	Score
NM_178135	3	260	L1MB4	448	714	c	0.8605	0.86	1597

```
CTGCTTTCTGTCTCTCTCTGTGTATTTGACTACTCTAGGTACCTCATATAAATGGAGTCATACAATATTT
||:||||||||||||----:|| ||||| |||:||||||||||||||||||:||||:||||||||||||
CTACTTTCTGTCTCT----ATGAATTTGMCTATTCTAGGTACCTCATATAAGTGGAATCATACAATATTT

ATACTTTTGCATCTGGCTTATTTCACTTAGCATAATGTCATTAAGGTTCATCTATCTAGTAGTATGTGTC
:| |||||:: |||||||||||||||||||||||||||||: |:||||||||||:|| | ||||:||||:||
GTCCTTTTGTGWCTGGCTTATTTCACTTAGCATAATGTTTTCAAGGTTCATCCATGTTGTAGCATGTATC

AGAATTTCCTTCCTTTCTAAGGCTGAGTAATATTCCATTGCATGTATATATCATATTTTGTTTATCT---
||||||||||||||||:||||||||:|||||||||||||:||||||||:||:|||||||||||:---
AGAATTTCCTTCCTTTTTAAGGCTGAATAATATTCCATTGTATGTATATACCACATTTTGTTTATCCATT

-----GTTGATGAACACTGGGGTTGTTCCCACTTCTTGGCTATTG-GAA----GTTGCTAT
-----|||||||:||||| |||||||:||||:|:||||||||||-|||----|:||||||
CATCCGTTGATGGACACTTGGGTTGTTTCCACCTTTTGGCTATTGTGAATAATGCTGCTAT
```

Fig. 16.4. CENSOR alignment between query sequence and its closest related Repbase consensus sequence. Summary data on the alignment are provided.

positions that produce positive scores in the alignment matrix. Finally, a summary table is provided listing the total number and length of elements identified from different Repbase classes and families.

3.2. RepeatMasker

RepeatMasker can also be run from a web server as shown in the following example. For consistency, the same 2-kb human promoter sequence (Genbank mRNA accession NM_178135) that was used in the CENSOR example is used for RepeatMasker. The URL http://www.repeatmasker.org/cgi-bin/WEBRepeatMasker points to the RepeatMasker submission page (**Fig. 16.5**). The first part of the query submission form is the "Basic Options" section. This allows for the uploading of a FASTA format sequence file or pasting sequence(s) into the text box. Users can select from a number of options that allow for the specification of the search algorithm to be used, the speed/sensitivity settings (*see* **Note 2**), the source of the query sequence, and the return format and method. More advanced search options can be found under the "Lineage Annotation Options" and "Advanced Options" sections (**Fig. 16.6**). The "Lineage Annotation Options" settings allow users to choose one or two comparison species against which the query can be surveyed for lineage-specific repeats. This can help to refine and narrow search results by eliminating elements common to many species. These options only work for mammalian query sequences due to the greater coverage of mammalian repeats in Repbase. The "Advanced Options" allow users to display alignments, choose how sequences are masked, decide on what kinds of repeats are to be masked, and adjust the sequence similarity matrix to be used based on GC-content of the query sequence.

RepeatMasker returns a number of output files describing the repeat content of the query and providing the masked sequence. A RepeatMasker summary file (or screen on the web server) is a

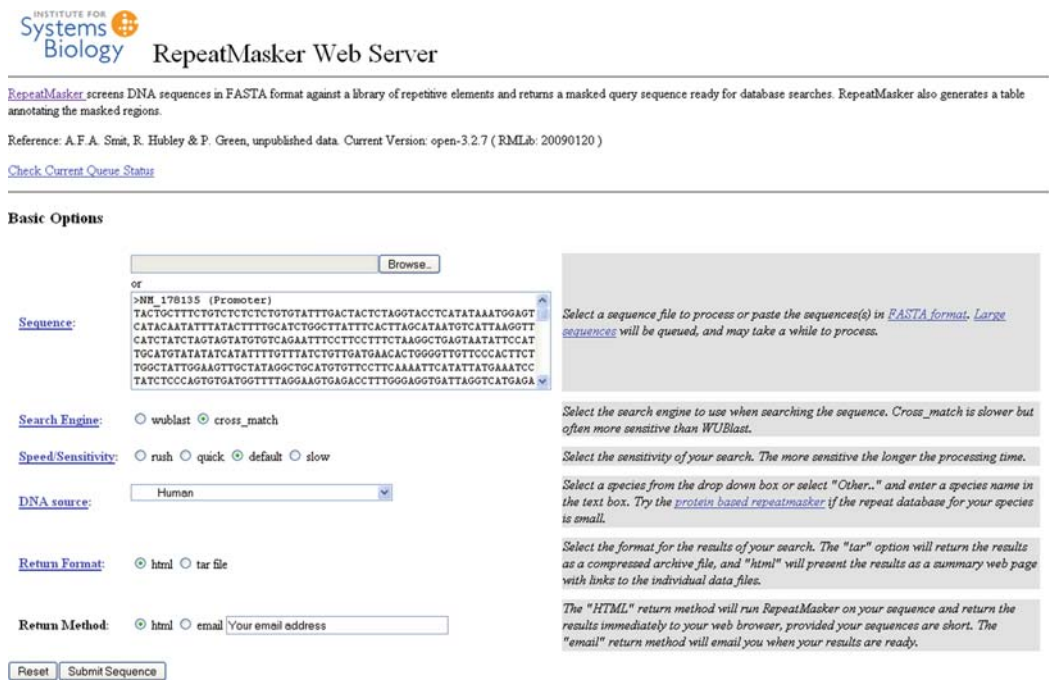


Fig. 16.5. RepeatMasker web server query submission page (Part 1). The "Basic Options" part of the submission page is shown.

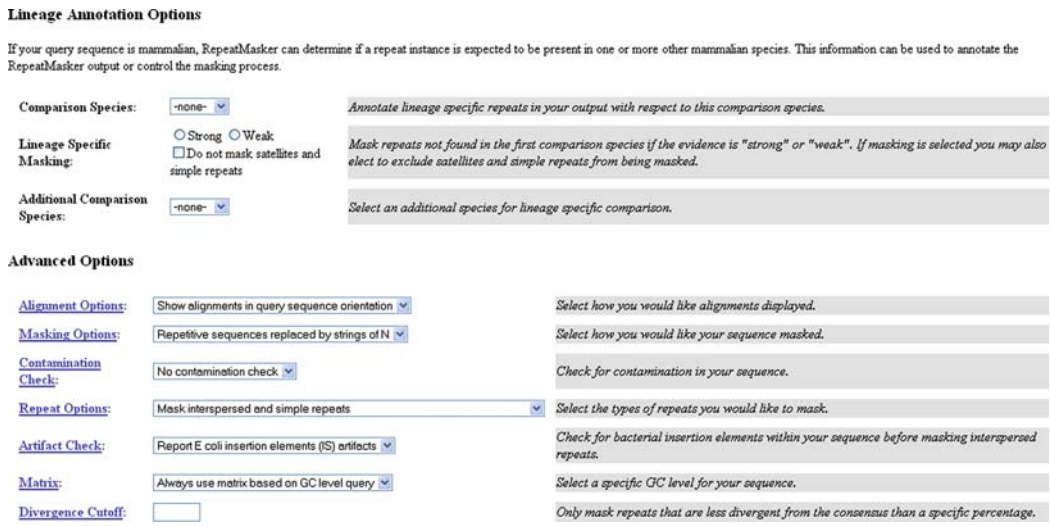


Fig. 16.6. RepeatMasker web server query submission page (Part 2). The "Lineage Annotation Options" and "Advanced Options" parts of the submission page are shown.

summary table that lists the percentage of query sequence masked by the different types of repeats (**Fig. 16.7**). A more detailed table is also provided with information on each individual repeat that is identified (**Fig. 16.8**). This table includes data on the

Summary:

```
==================================================
file name: RM2sequpload_1233250641
sequences:             1
total length:       2000 bp  (2000 bp excl N/X-runs)
GC level:          40.95 %
bases masked:        896 bp ( 44.80 %)
==================================================
                number of      length   percentage
                elements*    occupied  of sequence
--------------------------------------------------
SINEs:                 1          311 bp   15.55 %
      ALUs             1          311 bp   15.55 %
      MIRs             0            0 bp    0.00 %

LINEs:                 1          247 bp   12.35 %
      LINE1            1          247 bp   12.35 %
      LINE2            0            0 bp    0.00 %
      L3/CR1           0            0 bp    0.00 %

LTR elements:          1          338 bp   16.90 %
      ERVL             0            0 bp    0.00 %
      ERVL-MaLRs       1          338 bp   16.90 %
      ERV_classI       0            0 bp    0.00 %
      ERV_classII      0            0 bp    0.00 %

DNA elements:          0            0 bp    0.00 %
     hAT-Charlie       0            0 bp    0.00 %
     TcMar-Tigger      0            0 bp    0.00 %

Unclassified:          0            0 bp    0.00 %

Total interspersed repeats:       896 bp   44.80 %


Small RNA:             0            0 bp    0.00 %

Satellites:            0            0 bp    0.00 %
Simple repeats:        0            0 bp    0.00 %
Low complexity:        0            0 bp    0.00 %
==================================================
```

Fig. 16.7. RepeatMasker summary table output. Data on the length (bp) and percentage of different classes of identified repeats are provided.

SW score	perc div.	perc del.	perc ins.	query sequence	position in query begin	end	(left)		matching repeat	class/family	position in repeat begin	end	(left)	ID
1440	12.3	3.2	1.6	NM_178135	3	249	(1751)	C	L1MB3	LINE/L1	(216)	5967	5717	1
1514	19.1	8.9	0.9	NM_178135	255	592	(1408)	+	MLT1A0	LTR/ERVL-MaLR	1	365	(0)	2
1932	14.5	0.6	4.2	NM_178135	929	1239	(761)	C	AluJo	SINE/Alu	(12)	300	1	3

Fig. 16.8. RepeatMasker table output. Data for each individual repetitive element identified are provided.

levels of divergence between the query and consensus sequences along with location information specifying where the repeats are found in the query and which part of the repeats are represented. As was shown for CENSOR, RepeatMasker also provides a FASTA file with the repeats masked out, and the program can be configured to show alignments between repeats and their family consensus sequences.

RepeatMasker can also be run from the command line on Unix type operating systems. An example of the command line for the same search that was demonstrated for the web server is "RepeatMasker NM_178135.fasta -species human -alignments." Running RepeatMasker locally allows users to employ their own repeat libraries to search against. Another one of the advantages of the local RepeatMasker installation is the very detailed documentation that is provided including information on all command line options and flags. A list of all command line flags with brief descriptions can be obtained by simply typing "RepeatMasker" at the prompt. Typing "RepeatMasker –h(elp)" will print out all of the documentation.

4. Notes

1. Since CENSOR is powered by BLAST searches, search time varies directly with the length of the query and database and it can be run in three different speed/sensitivity settings. CENSOR uses WU-BLAST or NCBI-BLAST heuristics, which are both several times faster than the CROSS_MATCH dynamic programming algorithm employed as default by RepeatMasker.
2. The time complexity of the SW algorithm used by RepeatMasker is $O(n^2)$ where n is the word length. Therefore the time to process sequences increases sharply with length. Consequently, the speed settings are directly related to the word length used in CROSS_MATCH searches. In general, the program loses 5–10% sensitivity at each step of the speed settings, while gaining speed at a much higher rate. The time difference between the fastest and the slowest settings is approximately 30X. Heuristic WU-BLAST searches with RepeatMasker are generally much faster and compare to the fastest setting using CROSS_MATCH search algorithm.

Acknowledgments

The authors wish to thank Leonardo Mariño-Ramírez and Jittima Piriyapongsa for comments and technical support. Ahsan Huda and I. King Jordan are supported by the School of Biology at the Georgia Institute of Technology.

References

1. Lander, E. S., Linton, L. M., Birren, B., Nusbaum, C., Zody, M. C., Baldwin, J., Devon, K., Dewar, K., Doyle, M., FitzHugh, W., et al. (2001) Initial sequencing and analysis of the human genome. *Nature* **409,** 860–921.
2. Jurka, J. (2000) Repbase update: a database and an electronic journal of repetitive elements. *Trends Genet* **16,** 418–20.
3. Jurka, J., Kapitonov, V. V., Pavlicek, A., Klonowski, P., Kohany, O., and Walichiewicz, J. (2005) Repbase Update, a database of eukaryotic repetitive elements. *Cytogenet Genome Res* **110,** 462–7.
4. Jurka, J., and Milosavljevic, A. (1991) Reconstruction and analysis of human Alu genes. *J Mol Evol* **32,** 105–21.
5. Jurka, J., Walichiewicz, J., and Milosavljevic, A. (1992) Prototypic sequences for human repetitive DNA. *J Mol Evol* **35,** 286–91.
6. Jurka, J., Klonowski, P., Dagman, V., and Pelton, P. (1996) CENSOR – a program for identification and elimination of repetitive elements from DNA sequences. *Comput Chem* **20,** 119–21.
7. Kohany, O., Gentles, A. J., Hankus, L., and Jurka, J. (2006) Annotation, submission and screening of repetitive elements in Repbase: RepbaseSubmitter and Censor. *BMC Bioinformatics* **7,** 474.
8. Milosavljevic, A., and Jurka, J. (1993) Discovering simple DNA sequences by the algorithmic significance method. *Comput Appl Biosci* **9,** 407–11.
9. Smit, A. F. A., Hubley, R., and Green, P. (1996–2004) RepeatMasker Open-3.0 http://www.repeatmasker.org
10. Britten, R. J., and Kohne, D. E. (1968) Repeated sequences in DNA. Hundreds of thousands of copies of DNA sequences have been incorporated into the genomes of higher organisms. *Science* **161,** 529–40.
11. Morgulis, A., Gertz, E. M., Schaffer, A. A., and Agarwala, R. (2006) WindowMasker: window-based masker for sequenced genomes. *Bioinformatics* **22,** 134–41.
12. Bao, Z., and Eddy, S. R. (2002) Automated de novo identification of repeat sequence families in sequenced genomes. *Genome Res* **12,** 1269–76.
13. McCarthy, E. M., Liu, J., Lizhi, G., and McDonald, J. F. (2002) Long terminal repeat retrotransposons of Oryza sativa. *Genome Biol* **3,** RESEARCH0053.
14. McCarthy, E. M., and McDonald, J. F. (2003) LTR_STRUC: a novel search and identification program for LTR retrotransposons. *Bioinformatics* **19,** 362–7.
15. Rho, M., Choi, J. H., Kim, S., Lynch, M., and Tang, H. (2007) De novo identification of LTR retrotransposons in eukaryotic genomes. *BMC Genomics* **8,** 90.
16. Yang, G., and Hall, T. C. (2003) MAK, a computational tool kit for automated MITE analysis. *Nucleic Acids Res* **31,** 3659–65.
17. Quesneville, H., Bergman, C. M., Andrieu, O., Autard, D., Nouaud, D., Ashburner, M., and Anxolabehere, D. (2005) Combined evidence annotation of transposable elements in genome sequences. *PLoS Comput Biol* **1,** 166–75.
18. Gish, W. (1996–2004) WU-BLAST http://blast.wustl.edu
19. Altschul, S. F., Madden, T. L., Schaffer, A. A., Zhang, J., Zhang, Z., Miller, W., and Lipman, D. J. (1997) Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Res* **25,** 3389–402.
20. Green, P. (1994–1999) PHRAP and CROSS_MATCH http://www.phrap.org/phredphrap/phrap.html
21. Smith, T. F., and Waterman, M. S. (1981) Identification of common molecular subsequences. *J Mol Biol* **147,** 195–7.
22. Bedell, J. A., Korf, I., and Gish, W. (2000) MaskerAid: a performance enhancement to RepeatMasker. *Bioinformatics* **16,** 1040–1.

Chapter 17

DNA Sequence Polymorphism Analysis Using DnaSP

Julio Rozas

Abstract

The analysis of DNA sequence polymorphisms and SNPs (single nucleotide polymorphisms) can provide insights into the evolutionary forces acting on populations and species. Available population-genetic methods, and particularly those based on the coalescent theory, have become the primary framework to analyze such DNA polymorphism data. Here, I explain some essential analytical methods for interpreting DNA polymorphism data and also describe the basic functionalities of the DnaSP software. DnaSP is a multi-propose program that allows conducting exhaustive DNA polymorphism analysis using a graphical user-friendly interface.

Key words: Molecular population genetics software, DNA sequence polymorphisms, SNPs, Neutrality tests, Coalescent methods, DnaSP.

1. Introduction

The analysis of DNA sequence polymorphisms and SNPs (single nucleotide polymorphisms) can provide insights into the evolutionary significance of DNA polymorphisms, and into the selective and demographic factors acting on populations and species *(1–3)*. DNA polymorphism data, furthermore, are invaluable powerful tools (as molecular markers) in a wide range of disciplines such as biomedicine, animal and plant breeding, conservation genetics, epidemiology genetics, or forensics.

Modern high-throughput DNA sequencing and polymorphism detection methodologies are generating huge high-quality DNA sequence variation and SNPs data sets. This massive amount of data has stimulated the development of bioinformatics and analytical methods for handling, analyzing, and interpreting

David Posada (ed.), *Bioinformatics for DNA Sequence Analysis,* Methods in Molecular Biology 537

DOI 10.1007/978-1-59745-251-9_17

DNA polymorphism information. Current population genetics methods, and particularly those based on the coalescent theory, have become the primary framework to analyze DNA polymorphism data *(3, 4)*. Specifically, the comparative analysis of within-species DNA polymorphism and between-species variation is noticeably an effective approach to understand the evolutionary process and to obtain insights into the functional significance of genomic regions. In this context, the detection of both positive and negative selection is of major interest.

At present, there is a wealth of freely available computer programs for analyzing DNA polymorphism data (for a review *see* ref. *5*). Here, I explain the basic analytical methods implemented in DnaSP *(6)*; this software package is a multi-propose program that allows conducting exhaustive analysis using a graphical user-friendly interface. The main features of the software are as follows: (i) accommodates large data sets; (ii) computes many population genetic statistics describing the level and patterns of DNA polymorphism within and between populations; (iii) conducts computer simulation coalescent-based tests; (iv) generates graphical outputs rendering the information readily understandable.

2. Program Usage

2.1. Data Files

DnaSP accepts five input data file formats: FASTA, MEGA, NBRF/PIR, NEXUS, and PHYLIP *(6, 7)*. The data files should store a multiple DNA sequence alignment with polymorphism data (within-species variation), interspecific nucleotide variation (between-species variation), or any combination of both (*see* **Note 1**). Since all formats are in plain ASCII (text) files can, therefore, be viewed and edited in any plain-text editor. The software allows exporting and converting data files in the above mentioned formats, as well as in the format used by Arlequin *(8)* and NETWORK *(9)*.

2.2. Managing Data Information

Before conducting an analysis the data should be adequately prepared. For instance, to compute any statistic dealing with the number of synonymous substitutions, the user should specify the coding region positions, the genetic code, and the reading frame. DnaSP provides a user-friendly graphic interface (**Fig. 17.1**), where the user can specify these and many other features (codon preferences, genome type, ingroups and outgroups, chromosomal location, sites and sequence subsets, etc.). Certainly, not all analyses will require all data specifications; it is very convenient, however, to first define these features and save them on a NEXUS file format *(10)*. As the NEXUS format

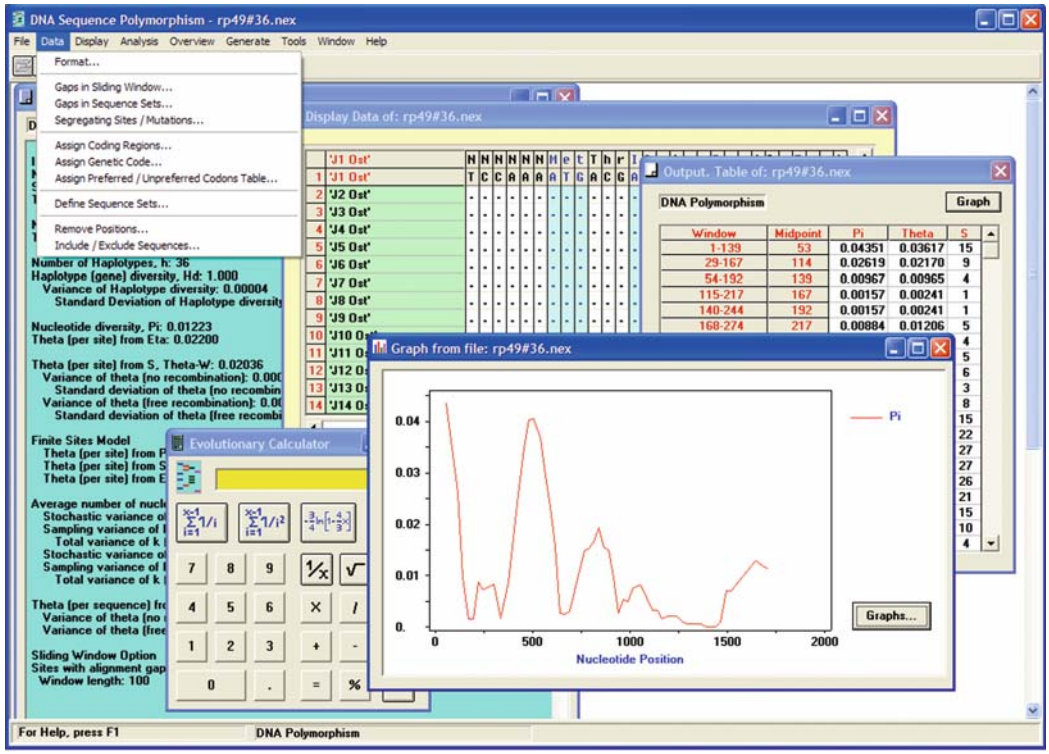


Fig. 17.1. DnaSP graphical user interface.

allows storing such information, the user no longer needs to define these data features again. Most importantly, data specifications written in DnaSP may be read by other population genetics and molecular evolution programs that make use of NEXUS files, such as MacClade *(11)*, PAUP*(12)*, and MEGA*(13)*; (*see* also ref. *5*) (*see* **Note 2**). DnaSP also includes a convenient DNA sequence browser, where the user can highlight alignment features such as polymorphic (variable) sites, singletons, parsimony-informative sites, invariant sites, synonymous and non-synonymous substitution sites, etc.

2.3. Analyses

The DnaSP software conducts exhaustive molecular population genetic analyses including those based on the coalescent theory *(4, 6)*. The software (i) measures the levels of DNA sequence polymorphism and SNPs (within and between populations), and divergence levels between species; (ii) estimates variation in synonymous and non-synonymous sites; (iii) analyzes the patterns of linkage disequilibrium, gene flow, and recombination; and (iv) computes the *P*-values of a number of neutrality tests using coalescent simulations. Furthermore, many of these analyses can be performed by the sliding window method, and

plotted. In this section, I describe some of the most representative and characteristic summary statistics and methods employed in the interpretation of DNA polymorphism data.

2.3.1. Summary Statistics

DnaSP computes most commonly used statistics quantifying the levels of DNA polymorphism *(14)*, such as the number of segregating sites, the average number of nucleotide differences, the number of haplotypes, and haplotype diversity, to analyze the distribution pattern of DNA variation, or to compare alternative evolutionary scenarios.

2.3.1.1. Site-by-Site Based Statistics

The number of segregating sites (S) is just the number of variable positions (i.e., polymorphic) in a sample of DNA sequences. Since the statistic does not utilize the information of the frequency of nucleotide variants, it is very sensitive to the sample size.

Nucleotide diversity (π), or the average number of nucleotide differences per site (i.e., the probability that two random sequences are different at a given site), is defined as

$$\pi = k/m \tag{1}$$

where m is the total number of nucleotide positions (including monomorphic positions, but excluding sites with alignment gaps) and k (often denoted as Π) is the mean number of nucleotide differences (*see* **Note 3**):

$$k = \frac{2}{n(n-1)} \sum_{i<j} d_{ij} \tag{2}$$

where n is the number of sequences (i.e., the sample size) and d_{ij} is the number of nucleotide differences between sequences i and j. The mean number of nucleotide differences can also be computed as

$$k = \sum_{i=1}^{m} h_i \tag{3}$$

where h_i is the heterozygosity at site i, which can be estimated as

$$h_i = \frac{n}{n-1} \left(1 - \sum_{j=1}^{4} x_{ij}^2 \right) \tag{4}$$

where x_{ij} is the relative frequency of nucleotide variant j ($j = 1, 2, 3$, and 4 correspond to A, C, G, and T) at site i. For large nucleotide diversity values ($\pi > 0.1$) it is convenient to apply the Jukes and Cantor multiple-substitution correction *(15)*. DnaSP allows computing these statistics primarily in the *Analysis|DNA Polymorphism* and *Analysis|DNA Divergence Between Populations* commands (*see* **Note 4**). DnaSP also provides a graphical representation of the site-frequency spectrum, i.e., the frequency distribution of segregating sites [*Analysis|Population Size Changes* command *Segregating Sites* option] (**Fig. 17.2**).

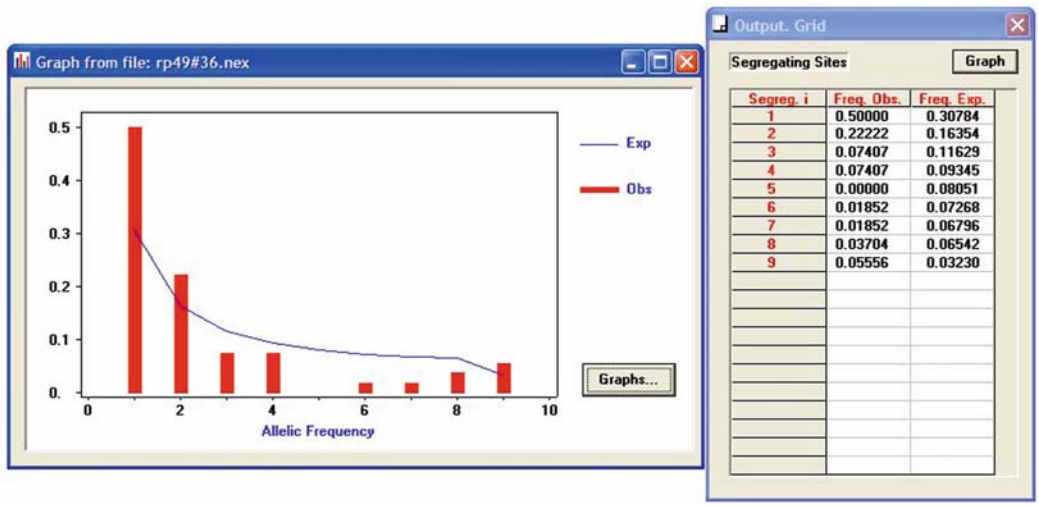


Fig. 17.2. Folded frequency spectrum of segregating sites.

2.3.1.2. Haplotype-Based Statistics

This category includes summary statistics that capture haplotype sequence information, for instance, haplotype diversity, linkage disequilibrium, and recombination statistics. Therefore, phased haplotype data is required. Hence, genotype (unphased) data – current routine SNP genotyping methods normally do not provide phase information – must be previously inferred to calculate these type of statistics; this step cannot be conducted with the current DnaSP version (*see* **Note 5**) and should be performed using other available algorithms and software *(16)*.

Haplotype diversity (*H*) – also known as gene/allele diversity or expected heterozygosity – is the probability that two random sequences are different, and is defined as

$$H = \frac{n}{n-1}\left(1 - \sum_{i=1}^{h} p_i^2\right) \tag{5}$$

where n is the number of sequences, h (often seen in the literature as K) the number of haplotypes (i.e., different DNA sequences), and p_i is the relative frequency of haplotype i. DnaSP calculates the haplotype diversity statistic in *Analysis|DNA Polymorphism*, *Analysis|Gene Flow and Genetic Differentiation*, and *Generate|Haplotype Data File* commands.

2.3.1.3. Mismatch Distribution-Based Statistics

This category includes descriptive statistics based on the mismatch distribution, namely the distribution of the number of differences between pairs of DNA sequences *(17,18)*. In spite of the popular nature and usefulness of these statistics in the analysis of past demographic events, they are very conservative, especially for recombining DNA regions *(19)*.

The raggedness *r* statistic *(20)* captures information of the shape of the mismatch distribution, which is affected by past population events. More specifically, *r* values differ between constant-size and growing populations. DnaSP provides a module to calculate *r* – and other commonly used statistics – [*Analysis*|*Population Size Changes* command], to estimate the relevant population parameters under a population growth scenario [*Model for Expected values* option] and to visualize the empirical and theoretical mismatch distributions (**Fig. 17.3**).

2.3.1.4. Neutrality-Based Statistical Tests

A neutrality test is a statistical method designed to test the neutral hypothesis, i.e., that all mutations are either neutral or strongly deleterious. There is a great variety of tests, which can be classified into different non-exclusive categories: (1) tests drawing information from synonymous non-synonymous substitutions (such as McDonald-Kreitman or d_N/d_S-based tests) *(21, 22)*; (2) tests that use only polymorphism (such as Tajima or Fu and Li's tests) *(23, 24)* or polymorphism and divergence data (such as Hudson, Kreitman, and Aguadé, or Fay and Wu tests) *(25, 26)* (*see* **Note 6**).

Here I will describe just one of the most popular DNA sequence-based statistical test, Tajima's *D* test *(23)*. Tajima's test employs polymorphism frequency spectrum data without taking into account synonymous and non-synonymous substitution information. Tajima's *D* statistic is defined as the standardized difference between two estimators of the population mutation rate parameter θ. For autosomal regions of diploid individuals $\theta = 4N\mu$, where N is the effective population size, and μ is the per-generation mutation rate. Specifically,

$$D = \frac{k - S/a_n}{\sqrt{Var(k - S/a_n)}} \tag{6}$$

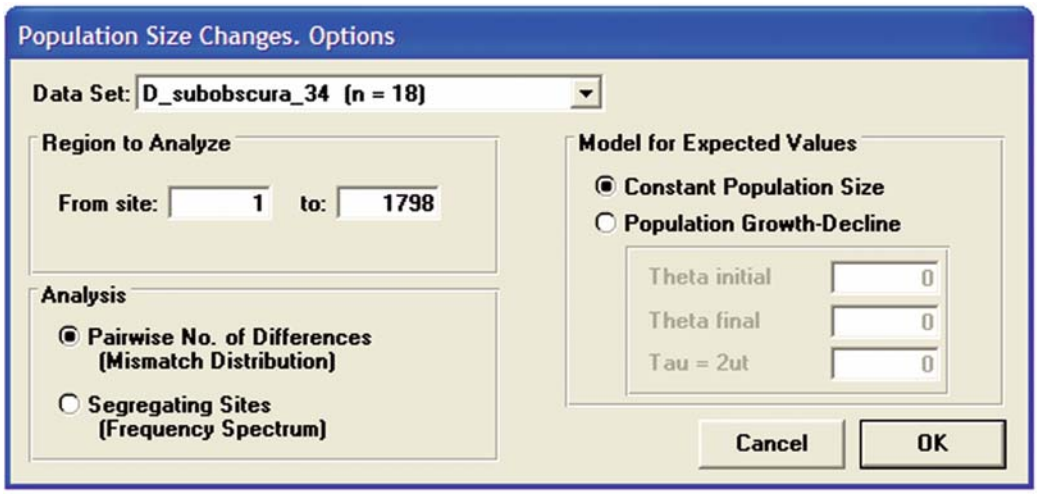
where

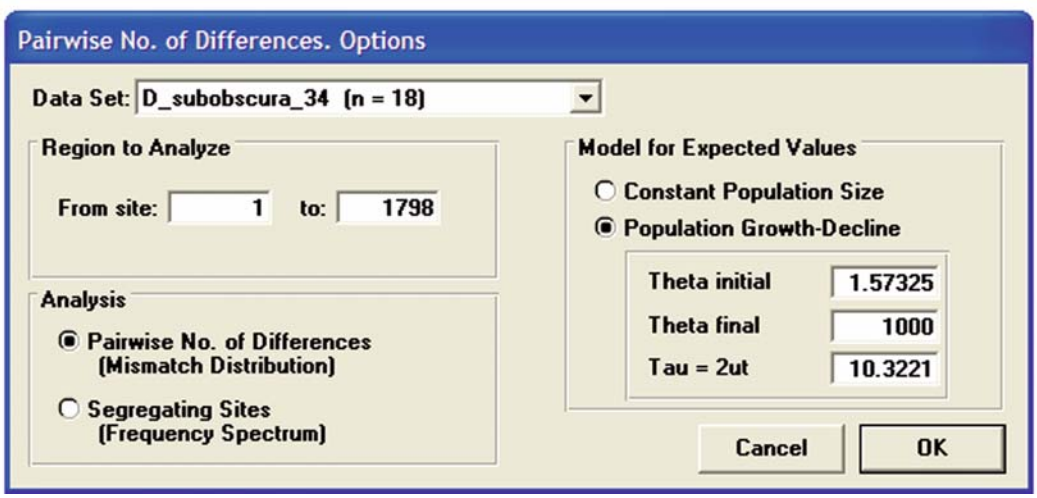
$$a_n = \sum_{i=1}^{n-1} \frac{1}{i} \tag{7}$$

Since under the standard neutral model the expected values of k and S are

$$E[k] = \theta \tag{8}$$
$$E[S]/a_n = \theta$$

the two estimates of θ (θ estimates provided by equations (8) and (9) are often symbolized as θ_π and θ_W, respectively) should provide roughly equal values and therefore D should be close to zero. To use this statistic as a test – particularly to contrast if a D value is significantly different from zero – it is necessary to know the





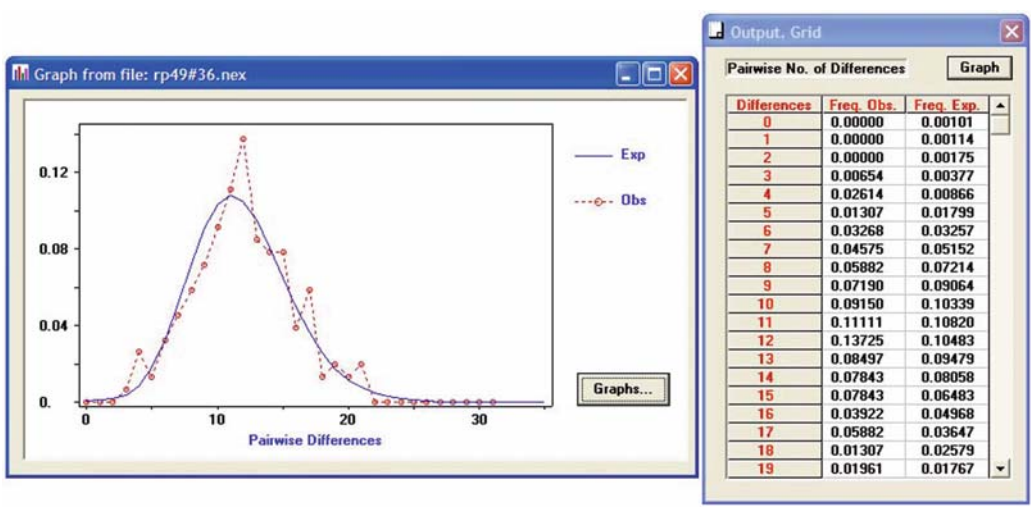


Fig. 17.3. Mismatch distribution plot. (**A**) Options for the first run. (**B**) Options for the second run. (**C**) Outcome results.

distribution of *D*. This distribution is usually obtained by computer simulations based on the coalescent process (see below). DnaSP provides specific modules to conduct these analyses [*Analysis*|*Tajima's Test* and *Tools*|*Coalescent Simulations* commands].

2.3.2. Sliding Window

The sliding window is a useful tool for locating genomic regions with atypical patterns of variation (*see* also ref. *27*); in DnaSP this method is implemented for a number of statistics [*Sliding Window*|-*Compute* option]. For instance, DnaSP can compute π values along a sliding window thus providing a graphical representation of the results (**Fig. 17.4**). Following equation (1), nucleotide diversity can also be calculated for synonymous and non-synonymous sites [*Analysis*|*Synonymous and NonSynonymous substitutions* command]; in addition, synonymous and non-synonymous variation within species (polymorphism) can be compared with between species (divergence) levels [*Analysis*|*Polymorphism and Divergence* command]. All these features are very useful for exploratory data analyses – such as detecting genomic regions with unusual patterns of variation – which, in turn, can provide insights into the functional constraints acting on genes or genomic regions.

2.3.3. Coalescent Simulations

The coalescent theory is a stochastic population-genetics model describing the statistical properties of gene trees *(4, 28)*. The coalescent provides very suitable methods for interpreting DNA polymorphism data – in fact it underlies the development of most neutrality tests– and for conducting efficient computer simulations. Indeed, the coalescent allows simulating samples under several different models (neutral, but also demographic or selective). These methods are essential for the detection of the signature of positive natural selection and for the distinction from the similar patterns generated by other (e.g., demographic) processes.

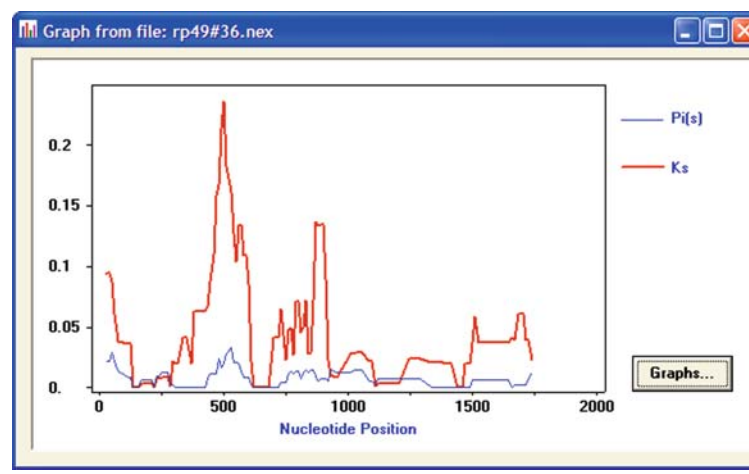


Fig. 17.4. Sliding-window plot of polymorphism and divergence levels.

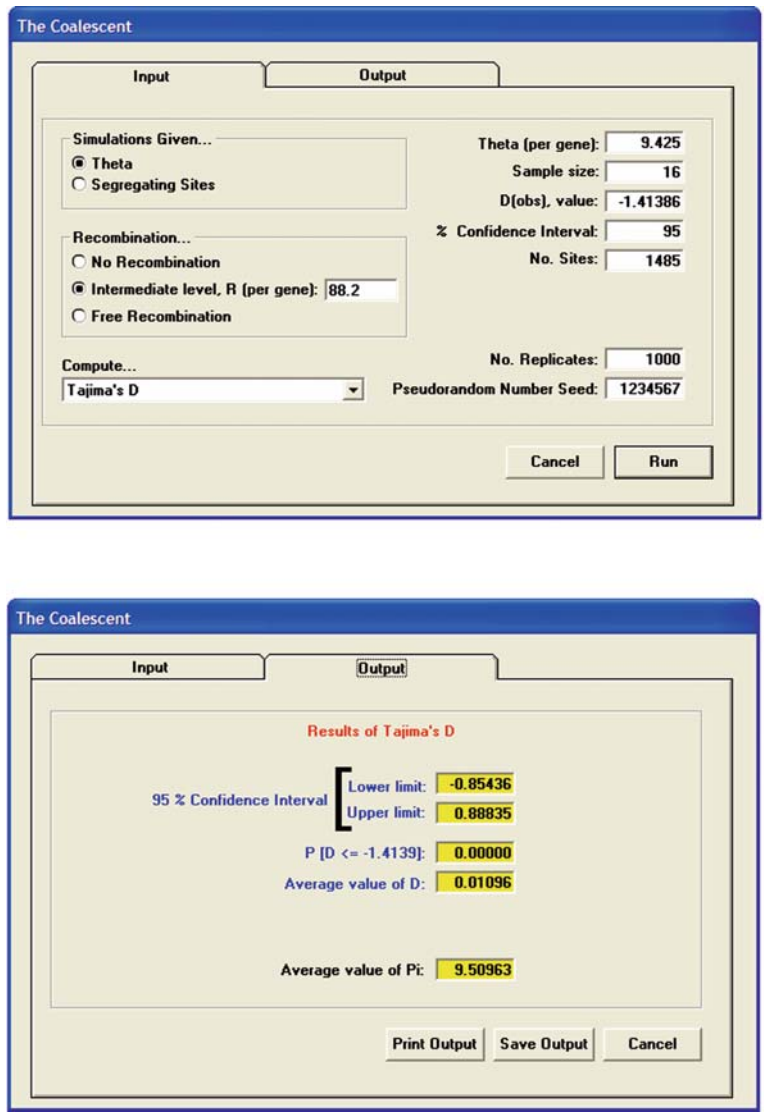


Fig. 17.5. Determining the confidence interval of Tajima's *D* by coalescent simulations. (**A**) Input Tab. (**B**) Output Tab.

DnaSP incorporates a specific module [*Tools*|*Coalescent Simulations*] for conducting coalescent-based computer simulations (**Fig. 17.5**). Gene trees are simulated under the null neutral coalescent model by applying different recombination rate values ($R = 4Nr$, where N is the effective population size, and r is the per-generation recombination rate between the most distant sites), and by fixing either the value of θ or the number of segregating sites S (*see* **Notes 7, 8**). Simulated samples are used to estimate the

empirical distribution of a wide variety of statistics, which, in turn, allow determining the confidence interval of the statistical tests, and conducting one-tailed or two-tailed tests (*see* **Notes 9, 10**).

3. Examples

The DnaSP software can be downloaded for free from http://www.ub.es/dnasp. The software package includes documentation (a windows help file) and some sample data sets. Here, I will use the rp49#36.nex file (data file distributed with the DnaSP software) as an example. The data set includes 36 multiple aligned DNA sequences of the ribosomal protein 49 gene region (~1.8 kb) from three species of Drosophila (ref. *29*). The data set contains four data subsets: D_subobscura_st (16 sequences of chromosomal arrangement O_{ST} from *D. subobscura*), D_subobscura_34 (18 sequences of chromosomal arrangement O_{3+4} from *D. subobscura*), D_madeirensis (one sequence from *D. madeirensis*), and D_guanche (one sequence from *D. guanche*).

3.1. Frequency Spectrum of Segregating Sites

Figure 17.2 shows a graph of the folded frequency spectrum of segregating sites [*Analysis*|*Population Size Changes* command *Segregating Sites* option]. The analysis was conducted using the D_subobscura_34 ($n = 18$) subset which contains $S = 54$ segregating sites. The *x*-axis of the graph represents the number of individuals carrying the least frequent nucleotide variant. The *y*-axis represents the proportion of segregating sites in the sample. For example, there are 12 polymorphic sites (12/54 = 0.2222) where the least frequent nucleotide variant is segregating in just two individuals. The expected values represent the values under the standard neutral model, i.e., a stationary constant-size population (ref. *23*; equation 50) (*see* **Note 11**).

3.2. Mismatch Distribution Analyses

The analysis was conducting using the D_subobscura_34 ($n = 18$) subset (**Fig. 17.3**). DnaSP generates the theoretical population growth distribution by a two run steps procedure [*Analysis*|*Population Size Changes* command]. On the first run (**Fig. 17.3A**) the software will estimate the relevant population growth parameters (*see* **Note 12**); in the second run (**Fig. 17.3B**) DnaSP will conduct the final analysis. The total number of pairs of sequences is $n(n-1)/2 = 153$ and the raggedness value $r = 0.0103$. The abscissa (**Fig. 17.3C**) gives the number of differences between pairs of DNA sequences (i.e., pairwise differences), and the ordinate gives the fraction of pairs that differ by that number. For instance, there are 14 pairs of sequences (14/153 = 0.0915; *y*-axis) differing by 10

differences (*x*-axis). The expected values are calculated assuming the sudden population growth model (ref. *18*; equation 4), and using the following parameter estimates: θ_0 = 1.573, θ_1 = 1,000, $\tau = 10.322$.

3.3. Sliding Window Analyses

Fig. 17.4 shows a sliding-window plot of the polymorphism (intraspecific data: *D_subobscura_st* subset) and divergence (interspecific data: *D_guanche* subset) [*Analysis*|*Polymorphism and Divergence* command]. The subset data file includes m = 1,485 sites (excluding alignment gaps). Polymorphism and divergence were expressed as π_S (silent nucleotide diversity per-site) and K_S (number of silent substitutions per silent site), respectively. Sites/Changes Considered option: *Silent* (*Synonymous & Noncoding*). Sliding window options: Window length = 50 bp; step size = 10 bp.

3.4. Coalescent Simulations

This example illustrate how to use DnaSP to determine the confidence interval of Tajima's *D* by computer coalescent simulations (**Fig. 17.5**) [*Analysis*|*Tajima's Test* and *Tools*|*Coalescent Simulations* commands]. The analysis was performed using the D_subobscura_st (n = 16) subset. This data includes m = 1,485 sites (excluding alignment gaps), being the observed Tajima's *D* value, D_{obs}= –1.41386. The computer simulations were conducted fixing the θ value (θ_π = 9.425), and considering recombination (R = 88.2); θ and R values are expressed on a per-sequence basis. The outcome (**Fig. 17.5B**) shows that the 95% confidence interval of Tajima's *D* lies between –0.85436 and 0.88835. Consequently, the observed *D* value is very unlikely under the standard neutral model (null hypothesis); indeed, the probability of obtaining values equal or lower than the observed $P[D \leq D_{obs}]$ is zero (*see* **Note 10**).

4. Notes

1. Although the DnaSP software has been designed to work with DNA sequence polymorphism data (including both monomorphic and polymorphic sites), it can also use data sets with only polymorphic positions (such as SNP haplotypes). In the latter case, however, not all analyses and methods will be applicable.
2. The NEXUS format is very convenient since it can store various types of information and can be read by many computer programs. This information, however, will be lost after exporting the data to simpler file formats.

3. Nucleotide diversity can be expressed on a per-site (π) or on a per-sequence (i.e., per-gene) basis (k). Many authors, however, use the same symbol (π) for both. The same is valid for θ, which is used either on a per-site or per-sequence basis.
4. DnaSP make most (but not all) estimates using the complete-deletion option. All multiple alignment columns with missing data or gaps are ignored.
5. DnaSP does not accept input files with genotype (diplotype) data. Such data should be previously converted to haploid phase information. Moreover, a number of analyses (i.e., haplotype-based analyses such as linkage disequilibrium or haplotype diversity) also require knowledge of the haplotype phase *(16)*.
6. Currently there is a high number of selective neutrality tests and test statistics. To determine which test should be used it is necessary to consider several factors such as the hypothesis that is being tested, the assumptions of the test, the study design, as well as the available information; in addition, the test should be chosen before observing the data. The choice of the test will depend on the specific reasonable alternative hypothesis; a statistical test performing well for detecting neutrality departures caused by hitchhiking events might be conservative against bottlenecks. Moreover, for a similar alternative scenario (e.g., population growth) may have different test statistics; in this case, the most powerful statistic should be chosen, i.e., the most powerful against a reasonable set of parameters of the alternative hypothesis (*see* ref. *19*).
7. The coalescent simulation module requires that θ and R values are expressed on a per-sequence (per-gene) basis.
8. Coalescent simulations can be conducted given a θ value (θ_π estimates) or by fixing the number of segregating sites. The former is recommended.
9. It should be stressed that a significant result (a significant departure from the null hypothesis) cannot be interpreted directly as evidence for positive selection; there are several putative alternative hypotheses to the single neutrality null hypothesis. For instance, the deviation may also be caused by demographic factors. Additional analyses, therefore, are required to determine the role of natural selection in shaping the patterns of nucleotide variation.
10. The specific *P*-value is obtained by comparing the observed (real data) test statistic value with the distribution of values (empirical distribution) obtained by computer simulations.

11. In the presence of an outgroup (unfolded frequency spectrum), the *x*-axis represents the number of individuals (i) carrying the derived nucleotide variant ($1 \leq i \leq n-1$). If there is no outgroup available, the frequency spectrum must be "folded" (folded frequency spectrum) to combine the indistinguishable categories (it is not possible to distinguish which nucleotide is ancestral and which is derived); in this case, the value $i = 1$ (*x*-axis) would indicate singleton configurations, i.e., where 1 individual (or $n - 1$ individuals) present a particular nucleotide variant, etc. In a folded frequency spectrum, therefore, $1 \leq i \leq n/2$.
12. DnaSP provides per-sequence estimates of the relevant population growth parameters: θ_0, θ_1, and τ; following Rogers *(30),* the estimates are obtained by fitting the empirical data to the theoretical expectations (method of moments).

Acknowledgments

We thank Sergios-Orestis Kolokotronis for his helpful comments and suggestions on the manuscript. This work was funded by grant BFU2007-62927 from the Dirección General de Investigación Científica y Técnica (Spain), and by grant 2005SGR00166 from Comissió Interdepartamental de Recerca i Innovació Tecnològica de Catalunya (Spain).

References

1. Przeworski, M., Hudson, R. R., and Di Rienzo, A. (2000) Adjusting the focus on human variation. *Trends Genet* **16,** 296–302.
2. Nordborg, M., and Innan, H. (2002) Molecular population genetics.*Curr Opin Plant Biol* **5,** 69–73.
3. Rosenberg, N. A., and Nordborg, M. (2002) Genealogical trees, coalescent theory, and the analysis of genetic polymorphisms. *Nat Rev Genet* **3,** 380–90.
4. Hudson, R. R. (1990) Gene genealogies and the coalescent process, in *Oxford Surveys in Evolutionary Biology* (Futuyma, D. J., and Antonovics, J. D., Eds.), Oxford University Press, New York. pp. 1–44.
5. Excoffier, L., and Heckel, G. (2006) Computer programs for population genetics data analysis: a survival guide. *Nat Rev Genet* **7,** 745–58.
6. Rozas, J., Sánchez-DelBarrio, J. C., Messeguer, X., and Rozas, R. (2003) DnaSP, DNA polymorphism analyses by the coalescent and other methods. *Bioinformatics* **19,** 2496–7.
7. Rozas, J., and Rozas, R. (1999) DnaSP version 3: an integrated program for molecular population genetics and molecular evolution analysis. *Bioinformatics* **15,** 174–5.
8. Excoffier, L., Laval, G., and Schneider, S. (2005) Arlequin (version 3): An integrated software package for population genetics data analysis.*Evol Bioinf Online* **1,** 47–50.
9. Bandelt, H.-J., Forster, P., and Röhl, A. (1999) Median-Joining networks for inferring intraspecific phylogenies. *Mol Biol Evol* **16,** 37–48.
10. Maddison, W. P., Swofford, D. L., and Maddison, D. R. (1997) NEXUS: an extendible file format for systematic information. *Syst Biol* **46,** 590–621.
11. Maddison, D. R., and Maddison, W. P. (2000) *MacClade 4*, version 4. Sinauer, Sunderland, MA.

12. Swofford, D. L. (1998) *PAUP*. Phylogenetic Analysis Using Parsimony (*and other Methods)*. Sinauer Associates, Sunderland, MA.
13. Kumar, S., Tamura, K., and Nei, M. (2004) MEGA3: integrated software for molecular evolutionary genetics analysis and sequence alignment. *Brief Bioinf* **5,** 150–63.
14. Nei, M. (1987) *Molecular Evolutionary Genetics.* Columbia University Press, New York.
15. Jukes, T. H., and Cantor, C. R. (1969) Evolution of protein molecules, in *Mammalian Protein Metabolism* (Munro, H. N. Ed.), Academic Press, New York, pp. 21–132.
16. Stephens, M., and Donnelly, P. (2003) A comparison of bayesian methods for haplotype reconstruction from population genotype data. *Am J Hum Genet* **73,** 1162–9.
17. Slatkin, M., and Hudson, R. R. (1991) Pairwise comparisons of mitochondrial DNA sequences in stable and exponentially growing populations. *Genetics* **129,** 555–62.
18. Rogers, A. R., and Harpending, H. (1992) Population growth makes waves in the distribution of pairwise genetic differences. *Mol Biol Evol* **9,** 552–69.
19. Ramos-Onsins, S. E., and Rozas, J. (2002) Statistical properties of new neutrality tests against population growth. *Mol Biol Evol* **19,** 2092–100.
20. Harpending, H. (1994) Signature of ancient population growth in a low-resolution mitochondrial DNA mismatch distribution. *Hum Biol* **66,** 591–600.
21. Mcdonald, J. H., and Kreitman, M. (1991) Adaptive protein evolution at the *Adh* locus in Drosophila. *Nature* **351,** 652–4.
22. Yang, Z., and Bielawski, J. P. (2000) Statistical methods for detecting molecular adaptation. *Trends Ecol Evol* **15,** 496–503.
23. Tajima, F. (1989) Statistical method for testing the neutral mutation hypothesis by DNA polymorphism. *Genetics* **123,** 585–95.
24. Fu, Y.-X., and Li, W.-H. (1993) Statistical tests of neutrality of mutations. *Genetics* **133,** 693–709.
25. Hudson, R. R., Kreitman M., and Aguadé, M. (1987) A test of neutral molecular evolution based on nucleotide data. *Genetics* **116,** 153–9.
26. Fay, J. C., and Wu, C. I. (2000) Hitchhiking under positive darwinian selection. *Genetics* **155,** 1405–13.
27. Hutter, S., Vilella, A, J., and Rozas, J. (2006) Genome-wide DNA polymorphism analyses using VariScan. *BMC Bioinf* **7,** 409.
28. Nordborg, M. (2001) Coalescent theory, in *Handbook of Statistical Genetics* (En Balding, D., Bishop, M., and Cannings, C., Eds.), John Wiley & Sons, Chichester, pp. 179–212.
29. Rozas, J., Segarra, C., Ribó, G., and Aguadé, M. (1999) Molecular population genetics of the *rp49*gene region in different chromosomal inversions of *Drosophila subobscura*. Genetics **151,** 189–202.
30. Rogers, A. R. (1995) Genetic evidence for a Pleistocene population explosion. *Evolution* **49,** 608–15.

INDEX

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

W

X